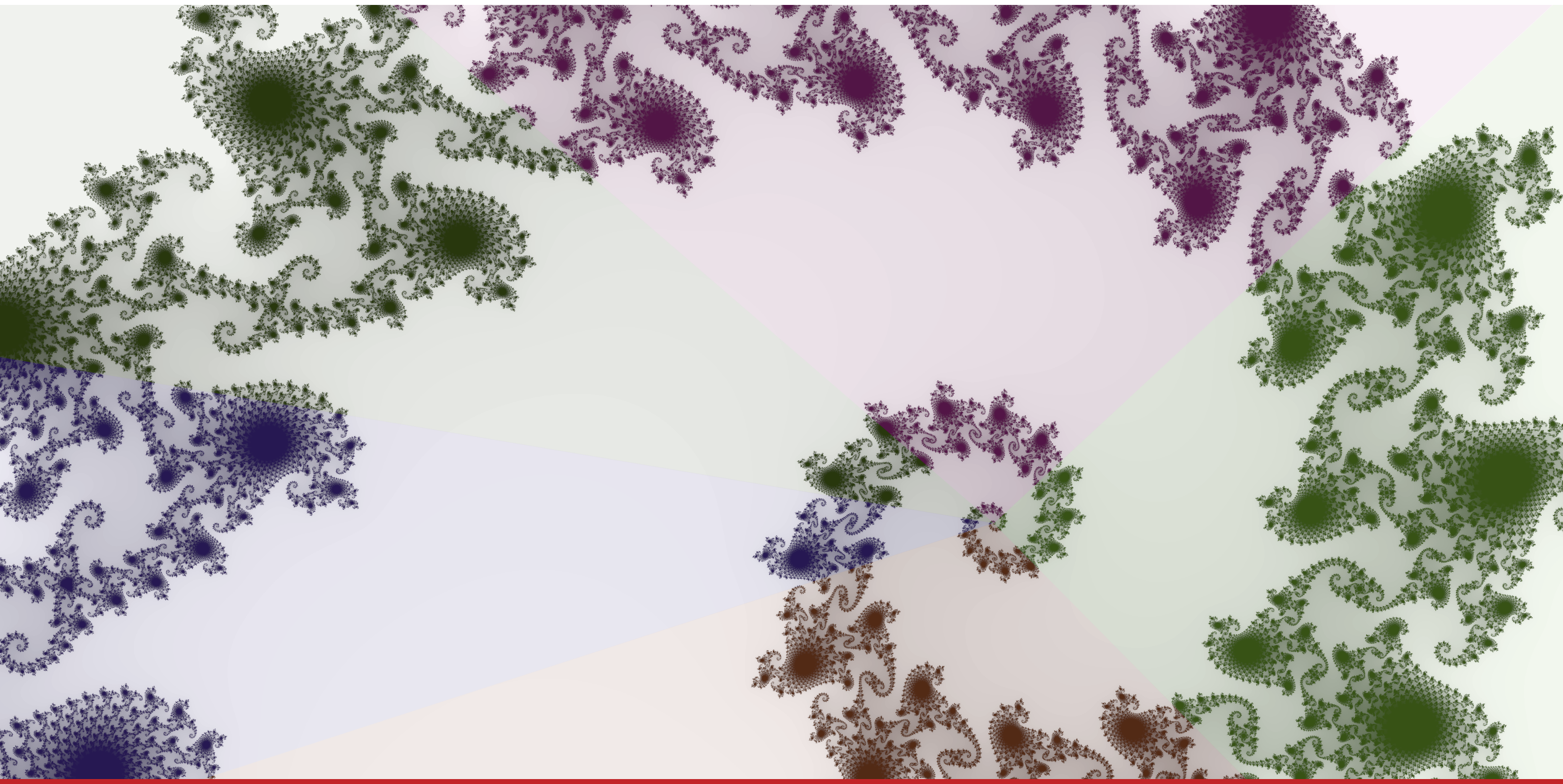
vorgelegt von

Müller, Jens, MSc.
aus Balingen

Tag der mündlichen Prüfung: ___________________



Robustness and Domain Generalization

Jens Müller

Supervisor
apl. Prof. Ullrich Köthe

Abstract

The robustness task asks to find a predictive model that remains accurate not only under known conditions but also when encountering completely novel situations, often referred to as distribution shift. Domain Generalization (DG) represents a specific case of robustness, wherein access to distinct environments is provided during the training phase. In this thesis, we contribute to the robustness task in four important ways. We propose two distinct frameworks in the context of DG designed to attain robustness. One is rooted in the fundamental principle of independent causal mechanisms, seeking invariances that persist across various environments. The second approach leverages contextual information about the origins of the data to improve robustness via permutation-invariant neural networks. Additionally, we define and implement a classifier's competence region as the region where the classifier is deemed competent and trustworthy. This approach enables us to increase the classifier's accuracy by rejecting samples lying outside its competence region, even in the presence of a distribution shift. Lastly, we theoretically analyze three crucial types of invariances, showing their prospects of success or failure in the robustness task under various distribution shifts. This comprehensive analysis provides a broad and insightful perspective on the topics of robustness and invariances.

Zusammenfassung

Die Aufgabe von Robustheit besteht darin, ein Vorhersagemodell zu finden, das nicht nur unter bekannten Bedingungen präzise bleibt, sondern auch in völlig neuen Situationen. Die Veränderung der Bedingungen wird auch als Verteilungsverschiebung bezeichnet. Domain Generalization (DG) stellt einen spezifischen Fall der Robustheitsaufgabe dar, bei dem während der Trainingsphase Zugang zu unterschiedlichen Bedingungen (oder Umgebungen) gewährt wird. In dieser Arbeit tragen wir insgesamt vier wesentliche Ansätze zur Robustheitsaufgabe bei. Wir schlagen zwei unterschiedliche Heransgehensweisen im Kontext von DG vor, die auf Robustheit ausgelegt sind. Eine basiert auf dem fundamentalen Prinzip unabhängiger kausaler Mechanismen und sucht nach Invarianzen, die in verschiedenen Umgebungen bestehen bleiben. Der andere Ansatz nutzt kontextbezogene Informationen über die Herkunft der Daten, um die Robustheit mithilfe von permutations-invarianten neuronalen Netzwerken zu verbessern. Zusätzlich definieren und implementieren wir den Kompetenzbereich eines Klassifizierers als den Bereich, in dem der Klassifizierer als kompetent und vertrauenswürdig angesehen wird. Dieser Ansatz ermöglicht es uns, die Genauigkeit des Klassifizierers zu erhöhen – selbst in Anwesenheit einer Verteilungsveränderung – indem wir Eingaben außerhalb seines Kompetenzbereichs ablehnen. Schließlich analysieren wir in der Theorie drei entscheidende Arten von Invarianzen und zeigen ihre Erfolgsaussichten oder ihr Versagen bei der Robustheitsaufgabe unter verschiedenen Verteilungsverschiebungen auf. Diese umfassende Analyse liefert eine breite und aufschlussreiche Perspektive zu den Themen Robustheit und Invarianzen.

Acknowledgments

First and foremost, I want to express my deep gratitude to my lovely wife, Carina, for her unwavering support during my PhD journey. Her encouragement and understanding have been a constant source of strength, allowing me to navigate through the challenges of this academic endeavor. Additionally, I am immensely thankful to my parents for their continuous encouragement, especially for helping us to care for our wonderful son, Lars, during this time.

I am also deeply grateful to my supervisor, Prof. Ullrich Köthe. Without his guidance, constructive feedback, and ideas, this thesis would not have reached its current level. Additionally, I want to express my gratitude to my co-supervisor, Prof. Carsten Rother, for his advice and constructive feedback throughout my doctoral journey. Furthermore, I would like to thank my second examiner, Prof. Artur Andrzejak, and the other members of the examination board, for their thorough examination and fair judgment of this thesis.

I would like to extend my gratitude to all my colleagues at the Computer Vision and Learning Lab (CVL) for the intellectual exchange and enriching discussions. I am especially thankful to Stefan Radev, Lynton Ardizzone, Jakob Kruse, Felix Draxler, Robert Schmier, as well as Martin Rohbeck. Furthermore, I would like to thank Martin Rohbeck, Felix Draxler, Robert Schmier, Stefan Radev, Peter Sorrenson, and Stefan Richter for proofreading parts of this thesis.

Finally, I wish to acknowledge the financial support provided by the Heidelberg Collaboratory for Image Processing (HCI) and the Informatics for Life program funded by the Klaus Tschira Foundation while working on this thesis. Additionally, I appreciate the Center for Information Services and High-Performance Computing (ZIH) at TU Dresden for its facilities for high-throughput calculations.

The title illustration depicts a colored fractal. Its inspiration draws loosely from seeking invariances in the robustness task: The fractal elicits certain invariant patterns that persist and the colors change across fields/environments. The majority of figures in this thesis have been drawn using TikZ and matplotlib. The icons that extend and illustrate some causal graphs are all drawn by my wife Carina. The layout design is taken and adapted from Jakob Kruse.

Contents

1 Introduction and Overview

Generalization is the fundamental task in machine learning [1, 2], and its significance extends to the realm of human intelligence [3–5], indicating its crucial role in the pursuit of Artificial Intelligence (AI). In the field of machine learning, generalization comes in two flavors: *out-of-distribution (OOD) generalization (or robustness)* and *standard generalization*[1] [6]. (Standard) Generalization refers to the ability to apply existing knowledge acquired in one set of circumstances to novel situations from the same set of circumstances. For predictive modeling, this is typically considered in the context of *supervised learning* [2]. Robustness or OOD-generalization in contrast refers to the situation where the circumstances under which knowledge is acquired differ from those in the encountered situations [6, 7]. The scenario where circumstances change is also termed *distribution shift*. In practical applications, predictive models are usually fine-tuned in one context, but might encounter situations in different contexts where they are expected to remain accurate [6, 8] – this corresponds to the robustness task. For instance, a tumor detector developed in one or several hospitals might also be applied in another one [9] or a model trained to analyze satellite images in one country should also work in another one [10]. Despite the practical relevance of the task and promising results [11], predictive models in AI and machine learning still lack strong (or human-comparable) robustness in many relevant challenges [12–14].

We contribute to the robustness task in four important ways. First, we relate different distribution shifts to a whole class of robust algorithms which allows us to understand when to expect failure or success. Second and third, we propose two distinct frameworks to improve robustness. One is rooted in causality and aims to identify invariances that promise robustness. The other uses contextual information about the data's origins to improve prediction under distribution shifts. Lastly, we define and implement the concept of a competence region that allows us to identify samples where a classifier can be considered trustworthy or not trustworthy.

1.1. Setting

Domain Generalization In the realm of machine learning and robustness, *Domain Generalization (DG)* depicts a particularly intriguing setting. The setup in DG involves having access to data from multiple environments (or domains) during training. The goal is to find a predictive model that performs well not only in the *seen* environments but also in entirely novel and unseen ones [15–17]. In the examples of tumor detection and satellite images from above, we might have access to images from different hospitals [9], or satellite images from different geographical zones [10] during training and fine-tuning.

[1]Standard generalization is henceforth referred to as generalization.

Deep Learning The goal of this thesis is to contribute to the robustness task in the context of *Deep Learning*. Deep Learning is a sub-field in machine learning (and AI), with its origins traced back to 1957 when Frank Rosenblatt introduced the perceptron [18]. The perceptron, loosely inspired by a single biological neuron, was emulated on an analog computer which was tediously slow at that time [19, Chapter 17]. The practical significance of deep learning, however, became evident much later, around the 2010s. It was during this time that advancements in computer power allowed for the scaling up of single perceptrons into larger deep neural networks consisting of millions or even billions of interconnected perceptrons, making them applicable to real-world scenarios [19, Chapter 17]. Notably, in 2012 deep learning models outperformed all comparing approaches in the challenging ImageNet competition [20]. Since then deep learning has been successfully applied in many fields, including natural language processing [21], neuroscience [22], protein structure prediction [23], and LHC physics [24]. We are currently experiencing a phase of great optimism in the field of AI arguably attributed to the deep learning revolution [19]. However, the field of AI is characterized by periods of optimism (summers) followed by unfulfilled hopes and disillusionment (winters) [19, 25]. Will deep learning share the same fate of an upcoming winter? In previous summers, expectations were not satisfied since they did not materialize in economic values leading to disappointed investors and loss of funding [25, Chapter 1.5]. This time is different: deep learning has already demonstrated its economic value. For instance, ChatGPT, a deep learning based product, reached over 100 million users within just two months [26]. Additionally, Yann LeCun, the head of Facebook's AI department (FAIR), has disclosed that the company is entirely built around deep learning [27]. Similarly, in other major tech companies like Google, deep learning is substantially integrated into a variety of products [28]. We can therefore conclude that deep learning and its successors are here to stay.

1.2. Approaches

Causality and Invariances Different (seen) environments that are accessible during training might indicate what knowledge remains applicable in all, potentially unknown environments. An important idea is to seek invariances that show across environments and exploit them for prediction [29–31]. For instance, we might have a dataset with cats in the environments cartoons, paintings, and images. In this case, the shape of the cat is invariant across environments and will be predictive in new environments (e.g., art pictures). In contrast, style elements vary across environments and do not promise robust predictions. In this work, we not only describe different types of invariances systematically (see Section 2.4), we also explain how different forms of invariances are grounded in causality (see Section 3.8). Furthermore, we elaborate on the relation between different types of invariances and distribution shifts, systematically answering the question of which type of invariance should be exploited for different distribution shifts to achieve robustness (see Section 3.8).

Learning robust models using the principle of ICM Stable causal relations induce an invariance (causal relation $\Rightarrow$ invariance) [30]. For instance, the shape of the cat *causes* it to be labeled cat and this relation is stable across different environments (e.g., art images and cartoons). To find a robust model we therefore need to find and exploit stable relations. We propose an objective that exploits invariances to deduce stable relations (invariance $\Rightarrow$ causal relation; see Chapter 5). This objective is derived from the principle of ICM [32, Chapter 2.1], a fundamental concept in causality that will be extensively discussed in Section 3.6. Importantly, we can prove that by optimizing this objective we indeed identify the underlying causal relations that remain stable across environments under suitable conditions. Our objective can be trained using gradient-based optimization and normalizing flows. Therefore, we avoid issues commonly encountered in discrete optimization, such as an exponentially growing search space, and address limitations that arise from inexpressive models, such as linear models. We demonstrate that models trained within our framework can identify the causal relations and exploit them for prediction, even when no inherently meaningful variables are provided, e.g. if only pixels are involved. This can be seen as a form of causal representation learning.

Context-Aware Domain Generalization Finding an invariance is in some scenarios unattainable (see Section 3.8) or not desirable. It could pose a disadvantage if we overlook the potential to enhance predictions by leveraging environment-specific information due to our focus on relying on some form of invariance. We propose an approach that exploits the characteristics of an environment to enhance the final prediction (see Chapter 6). With this approach, we can even withstand distribution shifts where it is impossible to find an invariance (see Section 3.8). We propose enhancing predictive models to be *context-aware*, adapting predictions to the current environment. The context of an input is represented by a set of i.i.d. samples originating from the same environment as the input itself. To integrate this "set-input" with standard models, we utilize permutation-invariant models to distill a summary embedding from the set-input. The set embedding acts as a condition for the inference network, which generates the final prediction (refer to Figure 6.5 for an overview). Considering a set representation has several advantages over a one-hot encoded environment label. First, we are not restricted to a pre-determined number of environments where the origin of inputs has to be always known. Second, our set embedding can represent relations between environments (e.g., closeness between environments). Third, discrete and continuous environment labels are equally informative for known environments, ensuring no loss of information.

We demonstrate empirically that this approach leads to improved predictions compared to baseline models on several datasets. In addition to establishing criteria that are crucial for achieving improvements, we prove their necessity theoretically and provide empirical demonstrations of how they can be validated using standard models. We also characterize the kind of distribution shift where our approach might yield benefits, namely the *source component shift*. As an example, consider the task of predicting housing prices based on factors like room size, building age, and proximity to the next shopping mall and school. In this case, the relationship between input factors and housing prices may vary due to location (e.g., housing prices in affluent areas tend to be higher), corresponding to the source component shift. When training our model on data from a couple of regions and applying it in a novel region, it could take advantage of contextual information, such as the properties of nearby houses, to adapt its prediction. Conversely, a simple baseline model lacking such context information would struggle in unfamiliar regions.

Additionally, we show how novel environments can be detected in the embedding space of our set-encoder. We employ this novel environment detection approach in two pivotal ways. First, we show that we can detect potential failure cases, for instance, due to entering an extrapolation regime with unknown prospects of success. Second, we show how the notorious trade-off between robustness to distribution shift and performance in the in-distribution (ID) setting [33, 34] can be overcome. This is achieved by selecting between the most robust and the most predictive model (with respect to the ID setting), according to whether a novel environment is detected.

Competence Regions In safety-critical applications such as autonomous driving, trading, or medical diagnosis mistakes can have severe consequences, making undetected failures (also called silent failures) highly problematic. In this thesis, we investigate inference with *failure detection* [aka *selective classification*, 35, 36] in the context of DG and classification. We introduce the concept of *competence regions* where a classifier can be deemed competent and trustworthy. Identifying samples that fall outside a classifier's competence region allows us to reject or delegate them to an expert, as in the case of medical applications, where a doctor's expertise can be consulted. Underlying to this approach is the fundamental trade-off between *coverage* and *accuracy*. By only retaining samples where the classifier appears highly competent, improves performance, at the price of reduced coverage. Conversely, maintaining high coverage comes at the cost of lower competence. To compute the competence region of a classifier we employ post-hoc OOD detection methods [37]. These methods utilize the feature or logit space of a classifier to identify anomalous instances. For instance, they may use a density estimation of the classifier's feature space and mark features that elicit sufficiently low density as OOD. The underlying idea of our work is to align the classifier's perception of OOD instances with its incompetence, hence making the choice of post-hoc OOD methods justifiable, given their utilization of the classifier for OOD detection. In our work, we investigate several different post-hoc OOD detection methods for computing the competence region and investigate various DG classifiers across 6 DG datasets. Additionally, we consider the closed (involving only known classes) vs. the open world (involving both known and unknown classes) scenario.

1.3. Contributions and Overview

Contributions We succinctly summarize our main contributions in this thesis as follows.

- We operationalize the principle of ICM making it amenable to gradient-based optimization (see Chapter 5).
    - We prove that under suitable conditions our method is able to uncover the underlying causal relations that promise robustness.
    - The use of normalizing flows allows us to generalize additive noise models, enabling the expression of more powerful functions.
    - We circumvent scalability issues known from combinatorial optimization.
    - We can empirically demonstrate that with our method we can perform causal discovery as well as causal representation learning (i.e. finding causal variables in high dimensional data, e.g., in pixel space).

- ▶ We propose a novel approach to Domain Generalization (DG) that leverages context information from new environments in the form of learnable set-representations (see Chapter 6).
  - We formalize the necessary and empirically verifiable conditions under which this approach can reap benefits from context information and improve on standard approaches.
  - We perform an extensive empirical evaluation and show that we can reliably detect failure cases when the necessary criteria of our theory are not met, or when extrapolation is required.
  - We demonstrate that we can detect novel environments which allows us to circumvent the notorious trade-off between being accurate in known environments and being robust to distribution shift.
- ▶ We introduce and investigate selective classification in the DG setting (see Chapter 7).
  - We systematically examine different post-hoc OOD detection methods to compute the competence region on various DG classifiers.
  - We observed that currently there is no satisfying way to determine the threshold that delineates competence from incompetence to ensure ID accuracy on OOD data, calling for further research.
  - We found that DG classifiers that aim to exploit domain knowledge do not exhibit a favorable competence region compared to a standard classifier.
  - We analyzed differences between post-hoc OOD detection methods in the open-world vs. closed-world setting.

In addition to these accomplishments, we also contributed in the following ways:

- ▶ We systematically relate different invariance properties with distribution shifts (see Section 3.8). This approach enables us to assess the potential success of various robustness algorithms based on the specific type of distribution shift.
- ▶ We propose ProDAS, a novel dataset that is an extension of the Dsprites dataset (see Chapter 6). ProDAS serves as a valuable playground for the investigation of various challenging domain shift scenarios (e.g., in Subsection 6.8.3).

Overview In the first three Chapters, we introduce the basics of robustness and domain generalization, causality, and deep learning. The focus here is three-fold. First, we introduce all the concepts necessary to understand our contributions in this thesis. Second, by introducing important concepts we relate them to robustness when possible. Third, we aim to systematically contextualize all concepts within a broader framework. In the causality chapter we rely heavily on [32] for definitions and concepts. In the deep learning chapter we use many definitions and views of [38].

In Chapter 3 we analyze which type of *invariances* might be beneficial according to the type of *distribution shift*. In Chapter 5 we present our approach on *learning robust models using the principle of ICM* that has already been published [31]. Chapter 6 introduces the *ProDAS dataset* [39] and presents our work on *context-aware DG* [40]. Afterwards, in Chapter Chapter 7 we present our approach on *finding competence regions in domain generalization* also published in [41]. In Chapter 8 we recap the most relevant aspects of this thesis, also pointing out intriguing research directions and discussing the broader perspective.

1.4. Publications underlying this thesis

Many of the substantial contributions in this thesis have been previously published, often in abbreviated formats due to space constraints. Next, I provide a list of the original articles, summarizing my contributions and providing references to the chapters where they are further elaborated in this thesis:

- **Learning Robust Models using the Principle of ICM** [31]
  Jens Müller, Robert Schmier, Lynton Ardizzone, Carsten Rother, and Ullrich Köthe
  DAGM German Conference on Pattern Recognition, 2021

  This article is adapted in Chapter 5. It constitutes predominantly my own and independent work, encompassing the initial idea and its conceptualization, developing the underlying mathematical framework, and executing the majority of experiments. My co-authors contributed to the writing process and creation of figures. Robert Schmier developed the architecture of the normalizing flow that we employ in the experimental part.

- **Finding Competence Regions in Domain Generalization** [41]
  Jens Müller, Stefan T. Radev, Robert Schmier, Felix Draxler, Carsten Rother, and Ullrich Köthe
  Transactions on Machine Learning Research, 2023

  This work is included in Chapter 7. The fundamental components of this article primarily stem from me. I initiated the project, developed the underlying concepts, and undertook the execution and analysis of the experimental segment. Collaboratively, my co-authors and I developed the mathematical framework, figures, and written content.

- **Towards Context-Aware Domain Generalization: Representing Environments with Permutation-Invariant Networks** [40]
  Jens Müller, Lars Kühmichel, Martin Rohbeck, Stefan T. Radev, and Ullrich Köthe
  arXiv preprint arXiv:2312.10107, 2023

  This work is adapted in Chapter 6. The initial idea to utilize permutation-invariant neural networks in Doman Generalization stemmed from Ullrich Köthe. I explored the idea and shaped it into the current form, including the underlying theoretical basics. I proved the mathematical results and did most of the experiments. Lars Kühmichel significantly contributed to the code infrastructure underlying the experiments and executed parts of an experiment himself.

- **ProDAS: Probabilistic Dataset of Abstract Shapes** [39]
  Jens Müller, Lynton Ardizzone, and Ullrich Köthe
  doi:10.11588/HEIDOK.00034135, 2023 and `https://github.com/XarwinM/ProDAS`

  This work is included in Chapter 6. I had the initial idea for this dataset and started to code an initial version. Lynton Ardizzone helped to extend the first version, both with ideas and code.

1.5. Notation, Terminology, and Basic Assumptions

Across this thesis, we will adhere to specific notational conventions aimed at enhancing the clarity and coherence of the mathematical content across various chapters. At certain parts, we will also introduce supplementary notation that aligns with the notation presented here, yet is relevant only to specific sections of this thesis.

The conventions we use are as follows:

X, Z, Y	a capital letter denotes random variables (RVs)
x, y, z	lower case letters denote realizations of a corresponding RV
$\mathbf{X}$ (or $\mathbf{x}$)	bold letters denote vectors or set of RVs (or realizations)
n	a lowercase letter denotes a scalar
$\mathbf{I}$	denotes the identity matrix
$\mathcal{F}$	denotes a set of functions
$P_{\mathbf{X}}$ or $P(\mathbf{X})$	is a probability distribution of RV $\mathbf{X}$.
$p_{\mathbf{X}}$	density of $P_{\mathbf{X}}$; RVs can be omitted for the sake of brevity.
	for discrete prob. measure, it denotes the prob. mass function
$P(Y \mid \mathbf{X})$	a collection of conditionals $P(Y \mid \mathbf{X} = \mathbf{x})$ for all $\mathbf{x}$
$I(X; Y)$	mutual information between X and Y
$I(X; Y \mid Z)$	mutual information between X and Y given Z
$H(X)$	entropy of X
$H(X \mid Y)$	entropy of X given Y
$\text{Var}(X)$	variance of X
$\text{Cov}(X, Y)$	covariance of X and Y
$\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$	is multivariate Gauss with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$

Table 1.1. Standard notation used throughout this thesis.

We assume that all RVs have a density p_A with probability distribution P_A. The independence and dependence of two variables A and B is denoted as $A \perp B$ and $A \not\perp B$ respectively. Two RVs A, B are conditionally independent given C if $P(A, B \mid C) = P(A \mid C)P(B \mid C)$. This is denoted with $A \perp B \mid C$ and it implies that A does not contain any information about B if C is known (see e.g., [32]). Similarly, one can define independence and conditional independence for sets of RVs. For simplicity, we assume throughout this thesis that the infimum and suppremum always exists in the corresponding reference set.

Expectations are denoted as $\mathbb{E}_{\mathbf{X}}$. If we want to emphasize that $\mathbf{X}$ follows a particular distribution P, we denote it as $\mathbb{E}_{\mathbf{X}\sim P}$ or $\mathbb{E}_{\mathbf{X}\sim p}$. ID denotes in-distribution, OOD denotes out-of-distribution, and i.i.d. stands for independent and identically distributed.

Robustness and Domain Generalization 2

In this chapter, we introduce and formalize the term *robustness*. We then consider one specific setting where robust models can be learned, namely *Domain Generalization (DG)*. In the setting of DG we introduce and discuss different types of invariances that may promise robustness.

2.1. Distributional Robustness

A *prediction task* asks to find a function (or predictive model) $f^\star$ that predicts a target Y from some input $\mathbf{X}$ as well as possible. For example, consider a hospital's desire to develop a function f, which can accurately predict the presence of skin lesions Y (represented as a binary variable) from a given skin image $\mathbf{X}$. In a more formal setting, we can define a prediction task in the context of *supervised learning*. The goal of classical supervised learning is to find a predictive model $f^\star$ that minimizes the prediction loss

$$f^\star = \operatorname*{arg\,min}_{f\in\mathcal{F}} \mathbb{E}_{(\mathbf{X},Y)\sim P}[c(f(\mathbf{X}), Y)] \tag{2.1}$$

on a distribution P. Here $\mathcal{F}$ is a family of functions (e.g., linear functions) and c is a task-specific loss or cost function (see e.g., Section 4.1). The solution $f^\star$ to the optimization problem in Equation 2.1 is, therefore, a best function out of $\mathcal{F}$ on the prediction task as presented by the distribution P and evaluated due to c. In Subsection 4.1.1 and Section 4.5 we discuss the supervised learning task in more detail.

In contrast, distributional robustness asks to find the best model $f^\star$ on a whole class of distributions $\mathcal{P}$ (see e.g., [7])

$$f^\star = \operatorname*{arg\,min}_{f\in\mathcal{F}} \sup_{P\in\mathcal{P}} \mathbb{E}_{(\mathbf{X},Y)\sim P}[c(f(\mathbf{X}), Y)] \tag{2.2}$$

Consequentially, $f^\star$ is the best predictive model in $\mathcal{F}$ on the worst case distributions $P \in \mathcal{P}$. In this work, we use the terms *distributional robustness* and *robustness* interchangeably. Note that the definition of the term "robustness" can vary between different research fields (e.g. our definition differs from the one in statistics) [7]. Furthermore, the robustness task as defined in Equation 2.2 is equivalent to the task of *out-of-distribution (OOD) generalization* [42].

Why is it necessary to consider a class of distributions in Equation 2.2 in contrast to just one distribution as in the supervised learning task in Equation 2.1? The main motivation to consider multiple distributions is that they offer a better description of many real-world problems. Often, when $f^\star$ is applied in a real setting, the encountered data samples can stem from multiple distributions. For instance, data encountered in different hospitals follow different distributions like in microscopy images of cells [9] or medical images of chest X-rays [43]. Reasons for the differences might include that devices collecting the data are different or simply because different hospitals treat different groups of patients. Also, satellite images can vary strongly across time and country/area [10]. It is typical, that during the optimization process, one has only access to a subset $\mathcal{P}' \subsetneq \mathcal{P}$ of all possible distributions in Equation 2.2. A model that has been optimized under specific circumstances (e.g. time range or physical location), may be inaccurate when conditions undergo alterations. This makes finding a solution to the prediction task particularly challenging and motivates the formulation of Equation 2.2 instead of Equation 2.1.

A particularly interesting application of robustness is *adversarial robustness* for deep neural networks. It is well-known that an attacker can deliberately add a small amount of noise $\mathbf{N}$ to the input $\mathbf{X}+\mathbf{N}$ such that the actual output Y remains unchanged, but common predictive models now predict a different and false Y [44]. This is mostly discussed in the context of image classification where a small amount of noise is added to the image. The class observed in the image does not change from a human perspective. But the previously correct deep neural network predicts now a completely different class. In adversarial robustness, one seeks a model f that still works even under distribution shift which is in this case adding small amounts of noise.

Solving the problem in Equation 2.2 becomes infeasible if distributions undergo arbitrary changes. Not only is it infeasible, but considering arbitrary distributions is also pointless from the perspective of a practitioner. In real-world applications, encountered distributions are typically related. In Section 2.4 we discuss different forms of invariances as a means to describe the family of distribution $\mathcal{P}$. We extend this discussion in Section 3.8 and show how causal models can be beneficial in describing families of distributions.

2.2. Domain Generalization

The main goal of this work is to contribute to the *robustness* problem as described in Equation 2.2 in the context of *Domain generalization (DG)*. In DG we term the distributions $P \in \mathcal{P}$ *environments* or *domains* and identify them with an index set $\mathcal{E}$, i.e., $\mathcal{P} = \{P^e \mid e \in \mathcal{E}\}$. Random variables with a superscript, e.g., Y^e, refer to a specific environment. We make the distinction between known environments $e \in \mathcal{E}_{\text{seen}}$, where training data are available, and unknown ones $e \in \mathcal{E}_{\text{unseen}}$, where we wish our models to generalize to. The set of all environments is then $\mathcal{E} = \mathcal{E}_{\text{seen}} \cup \mathcal{E}_{\text{unseen}}$. We can re-formulate the problem stated in Equation 2.2 in the context of DG as

$$f^\star = \operatorname*{arg\,min}_{f \in \mathcal{F}} \sup_{e \in \mathcal{E}} \mathbb{E}_{\mathbf{X}^e, Y^e}[c(f(\mathbf{X}^e), Y^e)] \tag{2.3}$$

The difficulty is that we need to solve Equation 2.3 only from $\mathcal{E}_{\text{seen}}$. The formula in Equation 2.3 is equivalent to Equation 2.2, but emphasizes the role of environments.

In this work, we assume that there is always more than one environment accessible during training, i.e. $|\mathcal{E}_{seen}| > 1$, but we like to mention that there is also work dedicated to the scenario where only one seen environment is assumed. This is called *single-source DG* [16]. In the DG classification literature, it is a common presumption that the set of

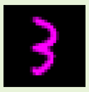
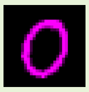
Data Set	Environments	Description/Infos
ColoredMNIST[34]	80%  90%  10%	A variant of the famous MNIST data set. Color-label association varies with the environment. Labels are noisy but consistently associated with shape.
PACS [46]	Art Sketch Cartoon Image	4 Environments with ≈6K images in total and 7 classes.
VLCS [47]	Caltech101 LabelMe SUN09 VOC2007	4 Environments with ≈10K images in total and 5 classes.
OfficeHome [48]	Art Clipart Product Real World	4 Environments with ≈16K images in total and 65 classes.
TerraIncognita [49]	Location 100 Location 38 Location 43 Location 46	4 Environments with ≈24K images in total and 10 classes.
DomainNet [50]	Clipart Infograph Painting Quickdraw Real Sketch	6 Environments with ≈600K images in total and 345 classes.

Figure 2.1. Overview of different DG datasets from the DomainBed repository [51]. All of these datasets are used in this work. For each dataset we show one class in various environments.

labels remains constant across various environments, and in our study, we uphold this assumption within the classification task. The scenario where the label space can change is referred to as heterogeneous DG [45]. For other variants of DG, see for instance [16]. In Figure 2.1, we present an overview of some interesting and relevant DG datasets.

2.2.1. Evaluation and Model Selection

The DG task asks to find a model that remains predictive when samples from a novel environment are presented. However, we do not know a priori how samples from this novel distribution might look, rendering the evaluation task initially unapproachable.

In literature, usually, one of three evaluation methods for model selection is employed: *training-domain validation*, *leave-one-domain-out cross-validation*, and *test-domain validation set* [16, 51]. In *training-domain validation* all seen environments are split into train and validation sets respectively. Different models and hyperparameter settings are then compared on the pooled validation set (i.e. pooled over all seen environments). In *leave-one-domain-out cross-validation* a model is trained on all seen environments except one left out for evaluation. This procedure is repeated so that we have an estimate for the expected loss on all left-out domains/environments. To aggregate these estimates, we can either average over all environments or use the estimate of the worst-case environment. It is common to repeat this whole procedure for different models and hyperparameter settings and choose the one where the best aggregated value (average or worst-case) is achieved. In *test-domain validation set* different models are trained on the seen environments and then compared on a validation set of the unseen test domain. The model that performs best is selected. This model-selection strategy is often used in literature but is actually not in line with DG where no knowledge about the unseen test environment is assumed.

While all these procedures seem justifiable, we do not have any guarantees for the model behavior in future, unknown environments. In real-world tasks, we usually do not know how the unknown environments might look, and therefore, model evaluation and model selection in DG are only accessible via a proxy measurement. It is also worth noting that the DG problem extends the complexity of the evaluation process. Given a dataset, the evaluation in a supervised learning task asks for the performance on novel data of the same distribution. Consequentially, one new sample indicates the performance of the model and since we typically have a test set, we can achieve reasonable estimates of the model's performance (see Section 4.5). In the context of DG, model evaluation faces an additional dimension of complexity due to the scarce amount of available environments and finite data therein, resulting in a very noisy performance evaluation. In the realm of DG, we can consider the environment as the object of interest ("How well is the performance for different novel environments?"). Since evaluations often involve just a few accessible environments (*environment scarcity.*), results might be very noisy and must be approached carefully.

2.2.2. Related Learning Settings

In various learning scenarios, we encounter data from multiple distributions and strive to address the prediction task. In the following, we discuss the most important ones. Closely related to Domain Generalization is *Domain Adapatation* (DA) [52]. In DA, we do not only have access to several environments during training but also to unlabeled/unvalued samples from the unseen test environment. Therefore, we know more about the test environment in DA compared to DG where we do not have access to any samples from the unseen test environment. The *Multi-task learning* setting uses the same information as DG during training, but asks for a less ambitious task: The test data are from the same environments as the seen environments. In *Transfer Learning*, we aim to find a model that works particularly well in one environment. Here, we have access to labeled/valued samples of the test environment. Often a model pre-trained on m environment is then fine-tuned for the new test environment. For a succinct comparison see Table 2.1 that is due to [51]. A more in-depth classification of DG and related fields can be found in [16] or [17].

Another interesting research field that is rarely considered in relation to DG is *multi-view learning*. In multi-view learning one has access to data from different views. Consider the example where a text is translated into different languages, then we have multiple views of the same content. In the DG context, the data would exhibit a distinct pattern: each text exists only in one language, but we possess also different texts written in other languages. It is easily seen that the multi-view learning setting offers more information than the DG setting. For more details on multi-view learning see [53].

Learning Setup	Training data	Test inputs
Domain Generalization	$L^1, \dots, L^m$	U^{m+1}
Multi-task learning	$L^1, \dots, L^m$	$U^1, \dots, U^m$
Domain Adaptation	$L^1, \dots, L^m, U^{m+1}$	U^{m+1}
Transfer Learning	$L^1, \dots, L^m, L^{m+1}$	U^{m+1}
Supervised Learning	L^1	U^1
Semi-supervised Learning	L^1, U^1	U^1

Table 2.1. Comparison between different learning setups. L^e and U^e denote the labeled and unlabeled datasets from environment e. The table is adapted from [51].

2.2.3. Methods – Overview

We can roughly categorize the DG methods into one of three categories: *data manipulation*, *representation learning*, and *learning strategy* [16]. In *data manipulation* the input distribution, where a model is trained, is extended either by *data augmentation* (e.g., [54, 55]) or by *data generation* (e.g., [56, 57]). In contrast, in the *learning strategy* category one aims to extend and adapt a common learning strategy to achieve improvements on the DG task. Methods could include for instance self-supervised learning (see e.g., [58, 59]). In self-supervised learning auxiliary tasks are formulated (e.g., image impainting, or predicting the rotation of a rotated image) in order to learn representations that might generalize beyond the seen environments. The *representation learning* category is the most popular in DG [16]. Methods in this category either aim to learn a representation of the inputs that elicit some form of invariance (see our work in Chapter 5) or aim to disentangle the learned representation/feature into domain-specific and domain-invariant parts to improve upon the DG task (e.g., [43, 60]). In Section 2.4 we discuss different forms of invariances. Our work in Chapter 5 falls in the category of *representation learning* via invariances. For more details on different approaches to DG see [16].

While it has been contended that DG methods do consistently outperform a simple baseline model trained via empirical risk minimization as observed in studies such as [14, 51, 61, 62], models trained on extensive datasets – comprising hundreds of millions of images – like CLIP [63] elicit highly generalizable features. These features are noteworthy, showcasing comparability to human-level abilities on certain tasks [11]. We applied the CLIP features to various DG datasets, resulting in the following outcomes in Table 2.2: A linear model fine-tuned on the CLIP features (denoted as Clip [Linear]), as well as a non-linear neural network applied to the CLIP features (denoted as Clip [Non-Linear]), significantly outperform the DG methods on PACS, VLCS, OfficeHome and DomainNet. In contrast, they underperform on TerraIncognita showing their generalizability might depend on the dataset at hand. For an overview of the underlying datasets see Figure 2.1.

2.3. Generative and Discriminative Models

Before delving into various kinds of invariances, we will briefly introduce two generic model types that significantly influence when an invariance might prove beneficial for the robustness task. In machine learning (for a brief introduction into machine learning see Section 4.1), it is commonly distinguished between *generative* and *discriminative models* (refer to, for instance, [68] or [8, Chapter 1.2]). *Generative models* aim to model the joint distri-

Algorithm \ Dataset	PACS	VLCS	OfficeHome	TerraInc	DomainNet
Clip [Linear]	**93.9** ± 0.1	79.7 ± 0.1	77.4 ± 0	29.8 ± 0.4	**51.7** ± 0.0
Clip [Non-Linear]	93.7 ± 0.3	**80.6** ± 0.2	**78.6** ± 0.2	36.9 ± 0.4	**51.7** ± 0.0
ERM [64]	85.5 ± 0.2	77.5 ± 0.4	66.5 ± 0.3	46.1 ± 1.8	40.9 ± 0.1
IRM [34]	83.5 ± 0.8	78.5 ± 0.5	64.3 ± 2.2	47.6 ± 0.8	33.9 ± 2.8
SagNet [65]	86.3 ± 0.2	77.8 ± 0.5	68.1 ± 0.1	**48.6** ± 1.0	40.3 ± 0.1
CORAL [66]	86.2 ± 0.3	78.8 ± 0.6	68.7 ± 0.3	47.6 ± 1.0	41.5 ± 0.1
MLDG [67]	84.9 ± 1.0	77.2 ± 0.4	66.8 ± 0.6	47.7 ± 0.9	41.2 ± 0.1

Table 2.2. Average accuracy and standard deviation of different algorithms on OOD data across various DG datasets, with each algorithm evaluated over 3 trials as in [51]. Model selection is conducted based on a validation set. Clip [Linear] denotes a linear model applied to the Clip features. Similarly, Clip [Non-Linear] represents a non-linear neural network applied to the fixed Clip features. All results, except the Clip-based models are sourced from [51].

bution $P_{\mathbf{X},Y}$ that describes the data [68]. Utilizing Bayes rule, the distribution of Y given $\mathbf{X}$ can be expressed as

$$P(Y \mid \mathbf{X}) = \frac{P(\mathbf{X} \mid Y)P(Y)}{P(\mathbf{X})} \tag{2.4}$$

In a binary classification setting where we aim to predict whether $Y = 0$ or $Y = 1$ for a given input $\mathbf{X} = \mathbf{x}$, we can compute the ratio:

$$\frac{P(Y = 0 \mid \mathbf{X} = \mathbf{x})}{P(Y = 1 \mid \mathbf{X} = \mathbf{x})} = \frac{P(\mathbf{X} = \mathbf{x} \mid Y = 0)P(Y = 0)}{P(\mathbf{X} = \mathbf{x} \mid Y = 1)P(Y = 1)} \tag{2.5}$$

This ratio does evidently not depend on $P(\mathbf{X} = \mathbf{x})$. To determine which class fits best with $\mathbf{x}$ we need to compute

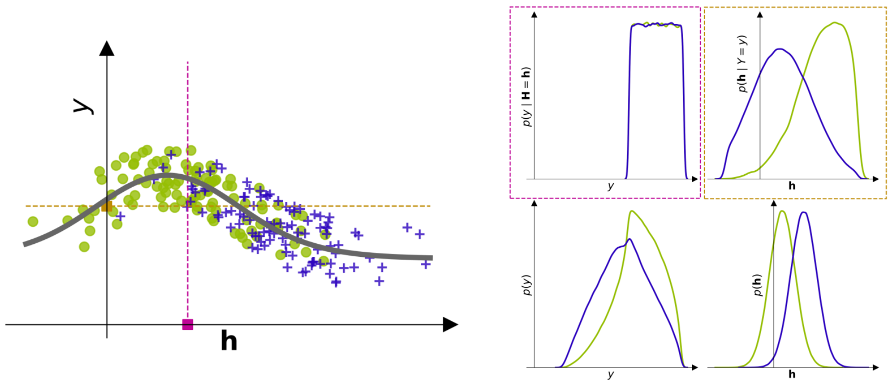
$$\arg\max_{y \in \{0,1\}} P(Y = y \mid \mathbf{X} = \mathbf{x}) \tag{2.6}$$

In simple words, we just need to determine whether the ratio in Equation 2.5 is $> 1, = 1$ or < 1. Hence, we can disregard $P(\mathbf{X})$ for the classification task. These considerations can be easily extended to classification problems with multiple classes.

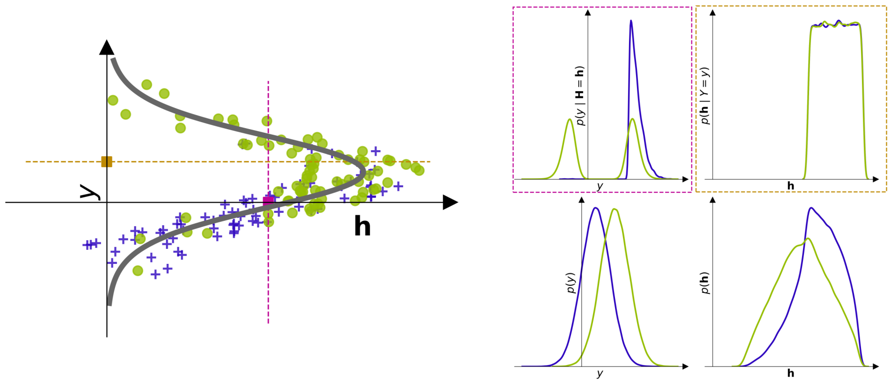
A *discriminative model* is just concerned with the prediction task of estimating $P(Y \mid \mathbf{X})$. Although it might seem unnecessary and cumbersome to consider generative models for prediction rather than discriminative models, there are specific advantages to considering generative models [68]. The distinction between discriminative and generative models has also important implications for the robustness task as discussed in both Section 2.4.

2.4. Invariances

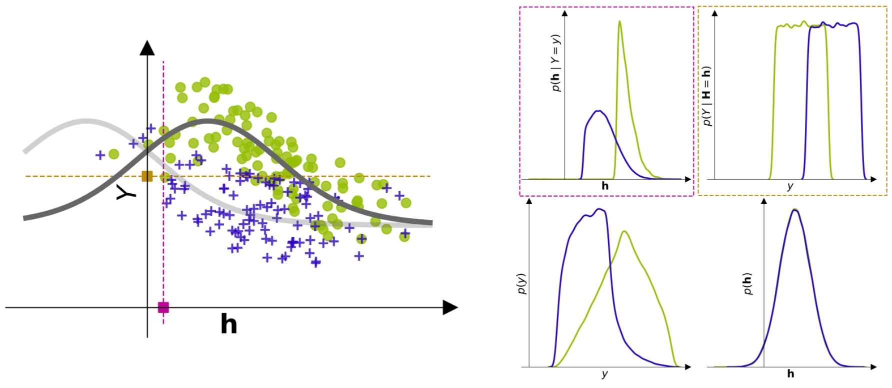
A mathematical object o is considered to be invariant under a transformation T if it remains unmodified under transformation T, i.e. it holds $T(o) = o$ [69]. In this thesis, we focus on a specific transformation – changing or shifting the domain or environment. The hope is that the invariance reflects an intrinsic property of the object that proves helpful



(a) *Causal invariance*: $h(\mathbf{X}) = \mathbf{H}$ follows a normal distribution and $Y = f(\mathbf{H}) + N$ where $N \sim U[-a, a]$ is uniformly distributed. Here, f is a non-linear function. The distribution of $\mathbf{H}$ varies between environments. It is evident from the ground truth model that $P(Y \mid \mathbf{H})$ remains invariant across both environments.



(b) *Anti-causal invariance*: Y follows a normal distribution in all environments. The features are $\mathbf{H} = f(Y) + N$ where f is some non-linear function and $N \sim U[-a, a]$) uniform noise. In this case, we can easily see that $P(\mathbf{H} \mid Y)$ is invariant across both environments.



(c) *Feature Invariance*: $\mathbf{H}$ follows the same marginal distribution in both environments. Additionally, we set $Y = f_e(\mathbf{H}) + N$ where $N \sim U[-a, a]$ is uniformly distributed and f_e represents a non-linear function that depends on the environment e. This formulation ensures $\mathbf{H} \perp E$.

Figure 2.2. All three types of invariances illustrated. *Left*: For each type of invariance we show data samples from two domains (one with green dots and one with blue plusses). *Right:* We also show the estimated prior of y and $\mathbf{h}$ as well as of the conditional distributions $p(y \mid h(\mathbf{X}) = \mathbf{h})$ and $p(\mathbf{h} \mid Y = y)$ at the marked points (magenta and orange).

for the robustness task. For example, consider the OfficeHome dataset in Figure 2.1, where the image style changes, but the shape, i.e. the intrinsic property of the object, remains invariant across environments.

Above, we defined the DG task and hinted at potential strategies to solve it (see Subsection 2.2.3). One crucial research field within DG addresses the DG problem by seeking a representation $h(\mathbf{X})$ of the input data $\mathbf{X}$ that exhibits an invariance property. The hope is that the invariance property enables accurate predictions in unseen environments. Therefore we adapt the non-accessible DG problem formulation in Equation 2.3 to

$$\underset{f\in\mathcal{F},h\in\mathcal{H}}{\arg\min} \sup_{e\in\mathcal{E}_{seen}} \mathbb{E}_{(\mathbf{X},Y)\sim P_e}[c(f(h(\mathbf{X})),Y)] \quad \text{s.t. } h \text{ satisfies invariance property} \tag{2.7}$$

This objective differs from Equation 2.3 in two crucial aspects. First, unlike a predictive model f operating directly on the input space, we consider a model f that operates on the representation $h(\mathbf{X})$, allowing us to impose constraints on the objective through an invariance property of h. Note that both f and h are optimized within Equation 2.7. Second, this objective is accessible during training since we only consider $\mathcal{E}_{seen}$ instead of all environments (including the unseen ones) $\mathcal{E}$. The hope is that the invariance property applies across all environments $\mathcal{E}$. While it might seem plausible that if the invariance property holds in $e \in \mathcal{E}_{seen}$, it should be valid in $e \in \mathcal{E}$, the concept of "invariance generalization" is a subject of debate, as discussed in [70]. Its validity may depend on the model class $\mathcal{F}$ and the specifics of the data distribution.

Different forms of invariances have been explored in the literature. Here, we introduce and discuss three important types, which we term *causal invariance*, *anti-causal invariance*, and *feature invariance*. In this thesis, we introduce a prefix before *invariance* to distinguish different manifestations. For an illustration of all invariances, refer to Figure 2.2. We begin with the *causal invariance*, also referred to as *invariance* henceforth. This specific form of invariance is the main focus in Chapter 5.

Definition 1 (Causal Invariance). *A feature $h(\mathbf{X})$ is **causal invariant** with respect to the environment E if the relation between $h(\mathbf{X})$ and Y is independent of the environment E, i.e. $Y \perp E \mid h(\mathbf{X})$. This is equivalent to $P(Y \mid h(\mathbf{X}), E) = P(Y \mid h(\mathbf{X}))$.*

Termed *conditional distribution alignment* in [71], this form of invariance is explored in our work in Chapter 5 and in other studies like [29, 30, 33, 71–73]. We illustrate this concept in Figure 2.2a. The causal invariance permits a change in $P(h(\mathbf{X}))$ as seen from the factorization of the joint distribution

$$P(h(\mathbf{X}), Y, E) = P(Y \mid h(\mathbf{X}))P(h(\mathbf{X}) \mid E)P(E) \tag{2.8}$$

Since $P(Y \mid h(\mathbf{X}))$ remains consistent across environments, it stays predictive even when encountering unknown environments. The scenario where only $P(h(\mathbf{X}))$ shifts is known as *covariate shift*[1]. We will delve deeper into covariate shift and other dataset shifts in relation to causality in Section 3.8. Another form of invariance is what we term *anti-causal invariance*:

[1]Covariate shift typically assumes $h(\mathbf{X}) = \mathbf{X}$ and $P(Y \mid \mathbf{X}) = P(Y \mid \mathbf{X}, E)$.

Definition 2 (Anti-Causal Invariance). *A feature $h(\mathbf{X})$ is called **anti-causal invariant** with respect to the environment E if $E \perp h(\mathbf{X}) \mid Y$. This is equivalent to $P(h(\mathbf{X}) \mid Y) = P(h(\mathbf{X}) \mid Y, E)$.*

For instance, this form of invariance has been investigated by [74–78]. We exemplify this type of invariance in Figure 2.2b. Similar to Equation 2.8, we can represent the joint distribution through a different factorization

$$P(h(\mathbf{X}), Y, E) = P(h(\mathbf{X}) \mid Y)P(Y \mid E)P(E) \tag{2.9}$$

Hence, we can conclude that the anti-causal invariance in Definition 2 allows for a change in $P(Y)$ as long as $P(Y \mid h(\mathbf{X}))$ is unaffected. The scenario where $P(Y)$ changes across environments, while $P(Y \mid h(\mathbf{X}))$ remains invariant, is often termed *target shift* [76, 77, 79]. In this case, however, predicting the probability $P(Y \mid h(\mathbf{X}))$ is not as straightforward as in the covariate shift example mentioned earlier. The difficulty arises from the fact that the conditional distribution $P(Y \mid h(\mathbf{X}))$ is given by:

$$P(Y \mid h(\mathbf{X})) = \frac{P(h(\mathbf{X}) \mid Y)P(Y)}{P(h(\mathbf{X}))} \tag{2.10}$$

and *both* $P(Y)$ and $P(h(\mathbf{X}))$ may vary across different environments. Given that $P(h(\mathbf{X}) \mid Y)$ is a generative classifier, $P(h(\mathbf{X}))$ can be disregarded. However, estimating $P(Y)$ may be impossible for unknown environments.

Another frequently examined form of invariance in the literature is what we term *feature invariance*.

Definition 3 (Feature Invariance). *A feature $h(\mathbf{X})$ is **feature invariant** with respect to the environment E if the feature $h(\mathbf{X})$ is independent of the environment E, i.e. $E \perp h(\mathbf{X})$*

This form of invariance is also referred to as *marginal distribution alignment* in [71] and has been explored in works such as [66, 80–84]. In Figure 2.2c we illustrate the feature invariance. Subsequently, we provide a practical example highlighting when this form of invariance leads to robust predictions.

If the feature-invariant representation lacks information about its originating environment, it implies that environmental factors (like various camera properties) have been removed from the representation $h(\mathbf{X})$. While such a representation might prove useful, it does not guarantee robustness on its own. Specifically, when $P(Y \mid h(\mathbf{X}))$ varies across different environments, as depicted in Figure 2.2c, issues arise. For example, the feature $h(\mathbf{X})$ might contain only shape information, omitting texture properties crucial for solving the task – like in certain skin cancer detection scenarios where texture information is indispensable.

In Section 3.8 we provide an in-depth discussion on the question of "when to use which invariance?". We can succinctly summarize the relation between robustness and predictiveness, i.e. $P(Y \mid h(\mathbf{X}))$, *and* the type of invariance in the following remark.

Remark 1. *As argued above, we achieve predictive guarantees in the case of the* causal *and* anti-causal invariance*, even under distribution shift:*

- *If we obtained a* causal invariant *feature* $h(\mathbf{X})$*, then* $P(Y \mid h(\mathbf{X}))$ *remains consistent across environments, ensuring its predictiveness.*
- *The* anti-causal invariance *can be exploited for prediction via*

$$P(Y \mid h(\mathbf{X})) = \frac{P(h(\mathbf{X}) \mid Y)P(Y)}{P(h(\mathbf{X}))} \tag{2.11}$$

In the regression setting, we are required to estimate how $P(h(\mathbf{X}))$ *and* $P(Y)$ *behave under distribution shift in order to robustly estimate* $P(Y \mid h(\mathbf{X}))$*. For classification, knowing how* $P(Y)$ *behaves under distribution shift suffices to estimate* $P(Y \mid h(\mathbf{X}))$*.*

However, feature invariance *does not guarantee predictiveness in new environments as illustrated in Figure 2.2c. Even if* $h(\mathbf{X})$ *follows the same distribution in all environments, i.e.* $h(\mathbf{X}) \perp E$*, the conditional distribution* $P(Y \mid h(\mathbf{X}))$ *might still unpredictably change across environments.*

It is worth noting that invariances have a close connection with fairness. For instance, a feature lacking information about sensitive variables such as age or gender is often deemed fair in an application process. Before we close this chapter, we briefly describe the difference between invariance and equivariance.

Remark 2 (Invariance and equivariance)**.** ***Invariance*** *and* ***equivariance*** *are specific properties of the relation between an object and a transformation. Invariance requires that the transformation has no effect on the object. For instance, consider bicycles in various environments (e.g. clipart, product, and art cartoons as in the OfficeHome dataset in Figure 2.1). The* ***shape*** *of the bicycles remains largely unchanged (or invariant) despite the distribution shift, whereas their* ***appearance*** *varies across the environment, making it not invariant. Equivariance primarily refers to a function or mapping, demanding that the transformation applied to the function's output is equal to the function applied to the transformed input. This concept can be formalized more mathematically: A mapping* f *is equivariant with respect to a transformation or a group of transformations* $g \in \mathcal{G}$ *if* $f(g(x)) = g(f(x))$*. In contrast,* f *is said to be invariant with respect to* $g \in \mathcal{G}$ *if* $f(g(x)) = f(x)$*. Convolution operations in neural networks exemplify equivariance, where a translation in the input corresponds to a translation in the output after the convolution operation [38, Chapter 9.2].*

Causality 3

Causality and robustness are closely related topics. In Chapter 2 we introduced the fundamental robustness task in Equation 2.2 which asks to find a good prediction even if the distribution shifts. Since this task is unsolvable in all generality, it is crucial to consider the characteristics of distribution shifts. Causality offers a perfectly suitable language for this task.

We begin this chapter by explaining the difference between causality and statistics, showing how statistical relations arise from causal relations. Next, we introduce both formal and conceptual aspects necessary to describe causal relations. Besides discussing different ways to formalize causal relations, we discuss the fundamental principle of independent causal mechanisms. Moving forward, we explore how distribution shifts can be characterized and categorized from a causal perspective, also discussing their implications and their relation to the invariances introduced in Section 2.4. We conclude this chapter by exploring methods and conditions for uncovering causal relationships.

3.1. Causality and Statistics

"Correlation is not Causation" is a commonly repeated phrase and basis for many hilarious observations as for instance the correlation between chocolate consumption and Nobel laureates per capita [86] *or* the correlation between drowning deaths and the appearance of movies starring Nicolas Cage (see Figure 3.1). While these observations seem ridiculous, they raise a serious question: How do *causation* and *correlation* relate?

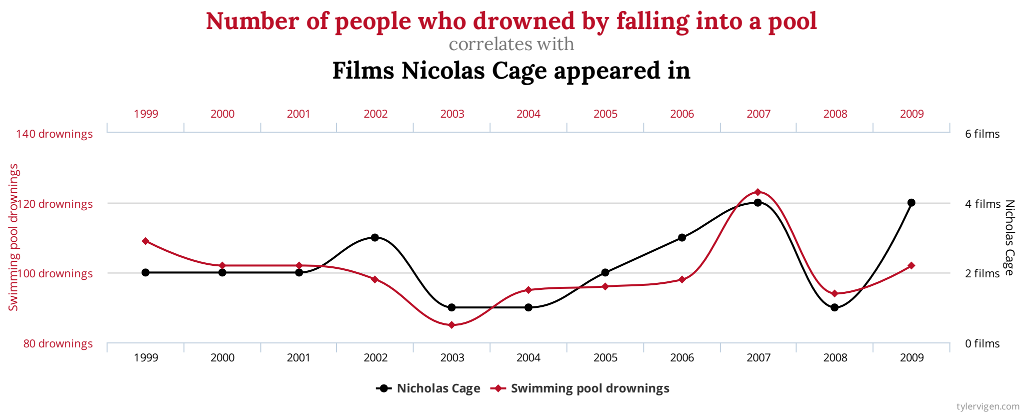


Figure 3.1. Correlation between *Number of people who drowned by falling into a pool* and the amount of *films starring Nicolas Cage* between 1999 and 2009. Graph due to [85].

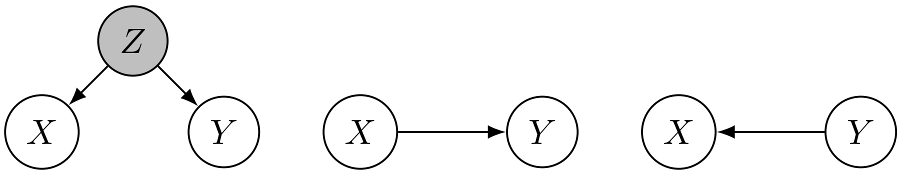


Figure 3.2. The common cause principle by Reichenbach gives three scenarios that explain a statistical dependency between two variables X and Y, i.e. $X \not\perp Y$.

First of all, we observe two insights: statistical dependency measures like correlation are *symmetric* while causal relations are *asymmetric*[1]. The first insight follows by definition of correlation and related statistical dependency measures. For the second insight, we consider the Nicolas Cage example from above. Whether Nicolas Cage appearances in movies cause people to drown and die *or* drowning people cause Nicolas Cage to appear in movies are two exclusive causal explanations. Conclusively, we can not fully deduce an asymmetric causal relation from a symmetric statistical one. However, we can ask which causal relations can be deduced from a statistical dependency or put slightly differently, "how do statistical dependencies arise from causal ones?". The famous *common cause principle* by Reichenbach provides an answer to this question [88]. As reformulated in [32, Page 7], the principle states:

Principle 1 (Reichenbach's common cause principle)**.** *If two random variables X and Y are statistically dependent ($X \not\perp Y$), then there exists a third variable Z that causally influences both. (As a special case, Z may coincide with either X or Y.) Furthermore, this variable Z screens X and Y from each other in the sense that given Z, they become independent, $X \perp Y \mid Z$.*

Figure 3.2 depicts all three scenarios listed in Reichenbach's common cause principle. In the chocolate-nobel-prize example from above, three possible causal explanations are (1) chocolate consumption could increase the consumer's intelligence and work ethic and therefore make it more likely to achieve a breakthrough research result, leading to a Nobel price, or (2) Nobel prize winners and their teams are notorious chocolate eater, increasing its countries chocolate consumption, or (3) there is a common cause for both like, for instance, a high Gross domestic product (GDP) might result in better education and therefore increase the likelihood for Nobel prizes, as well as higher chocolate consumption.

It is important to note here that we cannot distinguish purely from data and without assumptions which of the three common causes explains the statistical dependency between two variables (see Section 3.9 for details). Therefore we can conclude that a causal model contains strictly more information than a causal one (see also [89]). In Section 3.9 we discuss how it is still possible to learn about causal relations from data. While the common cause principle explains how statistical dependencies arise from causal relations, it does not explain all statistical dependencies. [32] hints at three reasons why dependencies might occur that cannot be explained by the common cause principle. We recap and extend on these reasons shortly.

In reality, we always deal with finite datasets and therefore, we might observe that some illusory dependencies would vanish if we collected more data. For instance, under the plausible assumption that the appearance of Nicolas Cage films is unconnected to the

[1]We do not consider cyclic causal relations in this thesis. For details on cyclic causal models see e.g. [87]

number of people drowning in pools, the correlation would disappear if Nicolas Cage appeared in enough movies over time. So this kind of correlation might be explained solely by the fact that we only observe a finite amount of data and not due to a common cause. Note that from a statistical point of view the variables would in these cases actually be independent and only the finite dataset let them appear dependent. As a second reason, two variables might exhibit statistical dependence without a common cause, provided they are not independently sampled. For instance, when we measure the same variables over time, they might both show some form of growth behavior. For instance, the shoe size of a child is correlated to the stock value of a company over time (both grow). However, assuming a causal relation between both variables might seem implausible. A third reason why statistical relations occur when there is no common cause involved is due to the *selection bias* which we discuss in Remark 5.

3.2. Graph Theory

It is instructive to think of causal relations in terms of nodes and edges. A node represents the variables of interest, such as GDP or chocolate consumption, while the edges depict how these variables relate. In general, two nodes are connected if one variable causes the other, and the direction of the edge indicates the causal relationship. Some examples of simple "causal graphs" are illustrated in Figure 3.2. In this section, we introduce the formal aspects of graph theory following [32, Chapter 6.1].

Consider a set of variables $X_1, X_2, \ldots X_D$. We define the set of nodes, denoted as V, as the set of indices corresponding to the variables, i.e. $V = \{1, 2, \ldots, D\}$. In the following, we may use the variable index interchangeably with the variable itself. For instance, we refer to variable i when we mean X_i and vice versa. Nodes can be connected by directed edges. Since cause-effect relationships are inherently asymmetric, we always assume a direction, leading us to mainly work with directed graphs. The set of edges $A \subset V^2$ is a subset of all tuples $V^2 = \{(i,j) \mid i, j \in V\}$. When we consider an edge (i,j), we refer to variable i as a parent of j, and j is as a child of i. This edge is also written as $i \rightarrow j$. The nodes V and edges A define the entire graph, denoted as $G = (V, A)$.

Sometimes we might be interested in the structure of G without the orientation of the edges. In such cases, we can consider the graph $(V, \widetilde{A})$ where $\widetilde{A}$ is defined as

$$\widetilde{A} := \{(i,j) \in V^2 \mid (j,i) \in A \text{ or } (i,j) \in A\} \tag{3.1}$$

This graph is also referred to as the *skeleton* of G and it reflects the presence of direct causal relations without specifying their direction.

Up to this point, we have primarily focused on direct connections in a graph, but we can extend our understanding to paths, which involve connections across more distant nodes. A sequence $i_1, \ldots, i_m$ of distinct nodes forms a path between i_1 and i_m if $i_{k-1} \rightarrow i_k$ or $i_k \rightarrow i_{k-1}$ for all $k = 2, \ldots, m$. If the path contains three consecutive nodes such that $i_{k-1} \rightarrow i_k \leftarrow i_{k+1}$, it is referred to as a *collider relative to this path*. This specific structure has significant causal implications and is discussed in greater length in Remark 5 and Section 3.9. When a path satisfies $i_k \rightarrow i_{k+1}$ for all $k = 1, \ldots m-1$, it is called a *directed path* from i_1 to i_m. In this case, we term all nodes $i_1, \ldots, i_{m-1}$ *ancestors* of i_m and all nodes $i_2, \ldots, i_m$ *descendants* of i_1. Ancestors of i are denoted by AN_i^G and descendants of i by DE_i^G.

In this work, we exclusively focus on directed acyclic graphs (DAGs), i.e. directed graphs that do not contain any *circles*. A circle consists of two directed paths: One from node j to k and another from k to j. We can define an ordering on a DAG in the sense that every node j that comes in the ordering before node k cannot be a descendant of node j. Orderings that satisfy this property are called a *causal orderings* or *topological orderings* and are defined as follows:

Definition 4. *Let G be a DAG. A permutation (i.e. a bijective mapping) $\pi\colon \{1,\dots,p\} \to \{1,\dots,p\}$ is called a causal ordering or topological ordering if it satisfies*

$$\pi(i) < \pi(j) \quad \textit{if } j \in DE_i^G \tag{3.2}$$

It is easily seen that for each DAG there exists a topological ordering (e.g., see [32, Proposition B.2]). The topological structure of a graph $G = (V, A)$ can be represented as an *adjacency matrix* A (here, we overload the variable notation A that also represents edges in G) which is a binary $D \times D$ matrix with

$$A_{i,j} = \begin{cases} 1 & \text{if } (i,j) \in A \\ 0 & \text{if } (i,j) \notin A \end{cases} \tag{3.3}$$

This implies that the entry (i, j) in the matrix is 1 if variable i causes variable j in G and otherwise 0. It is easily seen that for each DAG G there exists a topological ordering such that the corresponding adjacency matrix is an upper triangular matrix with $A_{\pi(i),\pi(j)} = 1$ if $\pi(i) > \pi(j)$ and $A_{\pi(i),\pi(j)} = 0$ elsewhere.

Judea Pearl introduced the d-separation criteria [90, 91] which plays a crucial role in identifying causal relations from observations (see also Section 3.9). d-separation is defined as follows

Definition 5 (d-separation)**.** *In a DAG $G = (V, A)$, a path between nodes i_1 and i_m is blocked by a set* **C** $\subset V$ *(with neither i_1 nor i_m in* **C***) whenever there is a node i_k, such that one of the following two possibilities holds:*

(i) $i_k \in$ **C** *and*

$$i_{k-1} \to i_k \to i_{k+1} \tag{3.4}$$
$$\textit{or } i_{k-1} \leftarrow i_k \leftarrow i_{k+1} \tag{3.5}$$
$$\textit{or } i_{k-1} \leftarrow i_k \to i_{k+1} \tag{3.6}$$

(ii) neither i_k nor any of its descendants is in **C** *and*

$$i_{k-1} \to i_k \leftarrow i_{k+1}. \tag{3.7}$$

Furthermore, in a DAG G, we say that two disjoint subsets of vertices **A** *and* **B** *are d-separated by a third (also disjoint) subset* **C** *if every path between nodes in* **A** *and* **B** *is blocked by* **C***. We then write*

$$\mathbf{A} \perp_G \mathbf{B} \,|\, \mathbf{C} \tag{3.8}$$

Below, we provide a short example of d-separation statements in a graph.

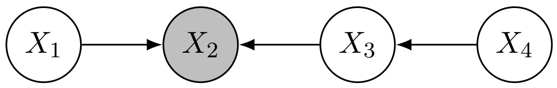
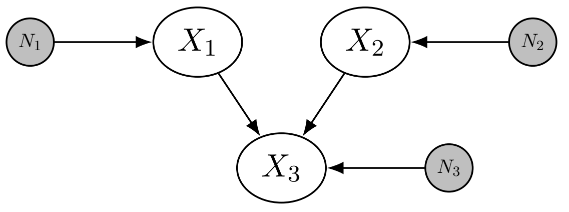


Figure 3.3. A simple example of a graph with four variables. In this graph, we can observe several interesting d-separation statements, e.g., $X_1 \perp_d X_3$.

Example 1. Consider the graph as in Figure 3.3. The collider structure $X_1 \to X_2 \leftarrow X_3$ implies $X_1 \perp_d X_3$ with the only path between X_1 and X_3 blocked by the empty set. Similarly, we obtain $X_1 \perp_d X_4$. However, if we condition on X_2, then the paths are no longer blocked. In this case, we write $X_1 \not\perp_d X_3 \mid X_2$ and $X_1 \not\perp_d X_4 \mid X_2$. However, if we condition on X_2 and X_3, then X_1 and X_4 are d-separated again, i.e. $X_1 \perp_d X_4 \mid X_2, X_3$. Similarly, the only path between X_4 and X_2 is blocked by X_3, i.e. $X_2 \perp_d X_4 \mid X_3$.

3.3. Structural Causal Models

We will first introduce and discuss the very basics of Structural Causal Models (SCMs). Afterward, we discuss their benefits, namely predicting interventions and answering counterfactual questions.



$$X_1 := f_1(N_1)$$
$$X_2 := f_2(N_2)$$
$$X_3 := f_3(X_1, X_2, N_3)$$

Figure 3.4. A simple example of an SCM with three variables as defined in Definition 6. The noise variables N_1, N_2, N_3 follow some distribution and are unobserved.

3.3.1. Definition

Structural Causal Models (SCMs) are a mathematical way to represent causal relations via functional relations. Following [32, Chapter 6.2] we define an SCM:

Definition 6 (Structural Causal Models)**.** *A **Structural Causal Model (SCM)** $\mathcal{S} = (\mathbf{S}, P_{\mathbf{N}})$ consists of a collection $\mathbf{S}$ of D **(structural) assignments***

$$X_j := f_j(\widetilde{\mathbf{X}}_{pa(j)}, N_j), \quad j = 1, \dots, D \tag{3.9}$$

where $pa(j) \subset \{1, \dots, j-1\}$ are called parents of X_j. $P_{\mathbf{N}}$ denotes the distribution over the noise variables $\boldsymbol{N} = (N_1, \dots, N_D)$ which are assumed to be jointly independent.

Succinctly, an SCM is described via the noise variables $N_1, \dots, N_D$ and the structural assignments $X_i := f_i(\mathbf{X}_{pa(i)}, N_i)$. The random variables X_j in an SCM correspond to nodes V in a graph, and the structural assignments $\mathbf{S}$ define the edges A in this graph, e.g. all nodes $pa(j)$ have an arrow pointing to j. Consequentially, an SCM defines a graph $G = (V, A)$ as for instance in Figure 3.4. This graph is *acyclic* due to the assumption that $pa(j) \subset \{1, \dots, j-1\}$. The parents $pa(j)$ are also called *direct causes* of X_j. The children of X_i in G are denoted as $ch(i)$ or $ch(X_i)$ and are often termed the *direct effects* of X_i.

It is easily seen that an SCM defines a unique distribution over the variables $\mathbf{X} = (X_1, \dots, X_D)$ [32, Proposition 6.3]. This is due to the acyclic definition of the structural assignments which implies that for each variable X_i we can find a function g such that $X_i = g(N_1, \dots, N_{i-1})$. This gives also a simple recipe on how to draw from the distribution $P_{\mathbf{X}}$. We first draw jointly from $N_1, \dots, N_D$ and then use the structural assignments to successively compute $X_1, \dots, X_D$. This shows that every SCM induces a unique distribution. Now, we raise the question of which distributions can be described via an SCM. So, we basically ask how powerful are SCMs? It turns out that SCMs can represent any distribution (see e.g., [32, Proposition 7.1] or Section 3.9):

Proposition 1. *Let $P_{\mathbf{X}}$ be a distribution that has a density with respect to the Lebesgue measure. Then there exists an SCM $\mathcal{S}$ that elicits the distribution $P_{\mathbf{X}}$.*

In Section 3.4 we will discuss in detail that the distribution $P_{\mathbf{X}}$ as defined via an SCM decomposes into the causal factorization

$$P(\mathbf{X}) = P(X_1, \dots, X_D) = \prod_{i=1}^{D} P(X_i \,|\, \mathbf{X}_{pa(i)}) \tag{3.10}$$

In the following remark adapted from [32, Section 6.2], we will show that several graphs can correspond to the *computationally* same SCM. In this thesis, we will therefore assume *structural minimiality* on SCMs to enforce a one-to-one correspondence between SCMs and graphs. Structural minimality is a formal assumption and will be introduced in the following remark.

Remark 3 (Structural minmality of SCMs)**.** *The following two SCMs are computationally equivalent and produce the same distribution $P_{X,Y}$*

$$\mathcal{S}_1 \colon X := N_X, Y := 0 \cdot X + N_Y \tag{3.11}$$
$$\mathcal{S}_2 \colon X := N_X, Y := N_Y \tag{3.12}$$

However, they correspond to two different graphs: one graph with an edge $X \to Y$ and one with no edge between X and Y. Therefore, we use the structural minimality assumption which requires all function f_i in the SCM to depend on all their input arguments. In a mathematical way, we can express this requirement as the following mathematical statement: If two structural assignments f_i and g_i produce the same output, i.e.

$$f_i(\widetilde{x}_{A_f}, n_i) = g_i(\widetilde{x}_{A_g}, n_i) \tag{3.13}$$

with indices set $A_f \subsetneq A_g \subset \{1, \dots i_1\}$, we choose f_i as structural assignment over g_i. Note that the structural minimality assumption does not pose assumptions on the reality of things, but induces a formal requirement that removes redundancies. In the example above, the SCM satisfying the structural minimality assumption guarantees a unique graph G that does not have an edge between X and Y.

We require in Definition 6 of an SCM that the noise variables are jointly independent. If an SCM indeed satisfies this assumption, we say the *causal sufficiency* assumption is satisfied. When this assumption is violated, then a hidden confounder H exists that is unobserved and affects two observed variables. We discuss implications of hidden confounders in Subsection 3.8.1.

3.3.2. Intervention

What makes SCMs particularly intriguing is their ability to not only model observational distributions but also to predict how the distribution changes under interventions or modifications. Our perspective on the robustness task in Equation 2.2 is strongly causality-oriented: We assume that distributions or environments change due to interventions. Predicting how a distribution behaves under interventions and modifications is therefore crucial for addressing the robustness task in Equation 2.2. Interventions, in a formal sense, correspond to modifications of one or several structural assignments in the SCM. We can define the interventional distribution similar to [32, Definition 6.8] as

Definition 7. *Let $\mathcal{S} = (S, P_{\mathbf{N}})$ be an SCM and its entailed distribution $P_{\mathbf{X}}$. An intervention is the replacement of one or several of the structural assignments in $\mathcal{S}$. Assume that we replace the assignment for X_k by*

$$X_k := \widetilde{f}_k(\mathbf{X}_{\widetilde{pa}(k)}, \widetilde{N}_k) \tag{3.14}$$

Note that $\widetilde{f}_k$ is different from f_k in the original SCM $\mathcal{S}$. We call the entailed distribution of the new SCM an interventional distribution. The modified structural assignment is said to be intervened on.

We can distinguish interventions based on the modification of structural assignments. For instance, if a structural assignment is set to a constant, i.e. $\mathbf{X}_k := a$ for a scalar $a \in \mathbb{R}$, it is referred to as an *atomic* intervention. When X_k is freed from all relations to other variables, i.e. $X_k = \widetilde{N}_k$, we call it a *hard* intervention. Note that we can similarly define interventions on the causal factors $P(X_j \mid \mathbf{X}_{pa(j)})$ of the causal factorization $P(\mathbf{X}) = \prod_{j=1}^{D} P(X_j \mid \mathbf{X}_{pa(j)})$. We expand on this in Section 3.6 and Subsection 5.3.1.

It is important to note here that correlation or predictiveness does not imply causation. In the following, we give an example of how predictiveness can lead to the wrong conclusion about causation.

Example 2. Consider the following linear Gaussian SCM that corresponds to G in the middle graph of Figure 3.7

$$X_1 \sim \mathcal{N}(0, 1) \tag{3.15}$$

$$X_2 = X_1 + N_2,\ N_2 \sim \mathcal{N}(0, \sigma_2^2) \tag{3.16}$$

$$X_3 = X_2 + N_3,\ N_3 \sim \mathcal{N}(0, \sigma_3^2) \tag{3.17}$$

We consider $Y = X_2$ (and $\sigma_Y = \sigma_2$) as target variable and assume that $\sigma_Y^2 = 1$ and $\sigma_3^2 = 0.01$. The correlations are Cov(X_3, Y) = Cov$(Y, Y) = \sigma_3^2 = 0.01$ and Cov$(X_1, Y) = \sigma_Y^2 = 1$. In this case the effect X_3 is more predictive than the cause X_1. In Example 6 we give a demonstrative example where children of target Y are more predictive than the parents of Y.

In Definition 7 we formally defined interventions in the SCM framework. Interestingly, we can also introduce interventions in the standard SCM setting by internalizing them as an environment variable. The following remark explains this approach and can be found similarly in [92, 93], [94, Chapter 3.2.2] or [32, Chapter 6.3].

Remark 4 (Interventions as Environments or Domains)**.** *Consider a scenario where we intervene on variable X_k. We can describe the intervention by replacing the structural assignment*

$$X_k := f_k(\mathbf{x}_{pa(k)}, N_k) \tag{3.18}$$

with a new structural assignment

$$X_k := \widetilde{f}_k(\mathbf{x}_{\widetilde{pa}(k)}, \widetilde{N}_k) \tag{3.19}$$

There is another way to describe this intervention that is often more convenient. We could introduce another variable E that indicates whether we have intervened upon X_k or not. Then we can define a new structural assignment

$$X_k := g_k('X_{pa(k)}, E, N_k) := \begin{cases} f_k(\mathbf{x}_{pa(k)}, N_k) & \textit{if } E = 0 \\ \widetilde{f}_k(\mathbf{x}_{\widetilde{pa}(k)}, \widetilde{N}_k) & \textit{if } E = 1 \end{cases} \tag{3.20}$$

From a graphical standpoint, we just added a variable E that is a parent of X_k. The node E acts as a source node within the SCM, lacking any parent nodes. In this scenario, we only consider one intervention, but we can easily extend this approach to arbitrarily many interventions.

The term *intervention* might insinuate that an intervention is done deliberately by an agent. However, interventions in our understanding can also be a description of differences between environments. For instance, if two image datasets follow the same distribution, except that different cameras recorded them. Let us assume that the camera that recorded the first dataset leaves some footprint in the images (e.g., adds Gaussian noise). Then we can describe the difference between the dataset as an intervention or via an environment variable (for details on this and similar examples see for instance Section 3.8).

It is important to note that ground truth functions in the structural assignments are typically unavailable and are often derived by fitting a function, $\hat{f}_k$, to predict X_k from $\mathbf{x}_{pa(k)}$, typically without considering interventions. Challenges arise when interventions occur, causing a shift in the support of the training data used to fit f_k. In such cases, extrapolation – that is unachievable without strong assumptions – might be necessary. We delve deeper into this phenomenon within the context of robustness in Subsection 4.5.4.

3.3.3. Counterfactuals

We have shown that SCMs are incredibly powerful, not only for modeling observational distributions but also for describing and predicting the outcomes of interventions. A third special feature of SCMs is their ability to answer counterfactual questions. A counterfactual question is a question of the type *what would have happened if we acted differently than we actually did.* This kind of question asks counter the actual facts. Since claims about a counterfactual world are never empirically falsifiable, they are sometimes contended as unscientific, as by the philosopher Karl R. Popper [95]. However, humans appear to use counterfactual reasoning in practice, acquiring this skill during their early childhood [96]. This indicates the relevance of counterfactual reasoning for artificial intelligence.

SCMs enable us to express and predict counterfactual questions. In contrast to interventional predictions, addressing counterfactual questions allows us to integrate future knowledge into our analysis. To illustrate this fact, consider the case where doctors have to decide what treatment to apply to a patient. While doctors only see the present information when asking which treatment to apply (interventional questions), they can obtain more information in the future, e.g., via an autopsy when the patient is dead. With more knowledge, they have a better understanding of how different treatments might have affected the patient (counterfactual questions). How do we exactly compute counterfactual questions in an SCM? This can be done via a three-step procedure as explained in [97]:

(i) Update the noise distribution according to the observed evidence (“abduction”)

(ii) Perform an intervention corresponding to the counterfactual question (“action”)

(iii) Predict the outcome with the modified SCM (“prediction”)

While interventional and counterfactual questions may seem alike at first glance, the key distinction lies in the type of information employed to answer these questions. To predict interventions, we can only use the knowledge available in the present moment. To answer counterfactual questions, we can also incorporate knowledge that might be obtained at some future point (abduction). We refer for details on counterfactuals to [32] or [94].

3.4. Markov Property and Faithfulness

In the following, we will introduce different concepts that connect a graph G with a distribution $P_{\mathbf{X}}$. These concepts are of importance, for instance, when we like to uncover the *true* causal graph G (e.g., the one that corresponds to the underlying SCM) from the distribution $P_{\mathbf{X}}$. We give an in-depth discussion of this task in Section 3.9.

Markov Property

The *Markov property* guarantees that we can read off conditional independence statements in $P_{\mathbf{X}}$ from the graph G. Equally to [29, Definition 6.21] we define it as

Definition 8 (Global Markov Property). *Let G be a DAG and $P_{\mathbf{X}}$ a joint distribution, then we say $P_{\mathbf{X}}$ satisfies the **global Markov property** with respect to the DAG G if*

$$\mathbf{A} \perp_d \mathbf{B} \,|\, \mathbf{C} \Rightarrow \mathbf{A} \perp \mathbf{B} \,|\, \mathbf{C} \tag{3.21}$$

for all disjoint set of nodes $\mathbf{A}, \mathbf{B}, \mathbf{C} \subset \mathbf{X}$. The symbol $\perp_d$ denotes d-separation as defined in Definition 5.

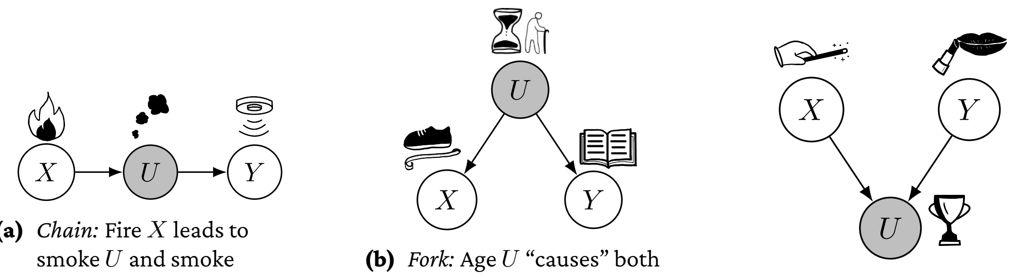
It can be shown that the *global Markov property* is equivalent to the *local Markov property* and the *Markov factorization property* [98, Theorem 3.27]:

Proposition 2. *If $P_{\mathbf{X}}$ has a density p, then the global Markov property with respect to G is equivalent to*

*(i) the **local Markov property** that requires that each variable in G is independent of its non-descendants given its parents, and*

*(ii) the **Markov factorization property** that requires the following factorization to hold*

$$p(\mathbf{x}) = p(x_1, \ldots, x_D) = \prod_{j=1}^{D} p(x_j \,|\, \mathbf{x}_{pa(j)}) \tag{3.22}$$



(a) *Chain:* Fire X leads to smoke U and smoke subsequently triggers the fire alert Y.

(b) *Fork:* Age U “causes” both shoe size X and reading skills Y.

(c) *Collider*: Talent X and beauty Y “cause” success U.

Figure 3.5. Illustrative examples for elementary causal structures. These graphs are considered in Example 3.

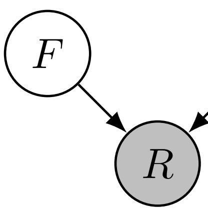
Due to this equivalence, we often speak only of *Markov property*. An important property of SCMs is that they naturally elicit the Markov property [94, Theorem 1.4.1]:

Proposition 3. *Assume that $P_{\mathbf{X}}$ is induced by an SCM with its underlying graph G. Then, $P_{\mathbf{X}}$ is Markovian with respect to G.*

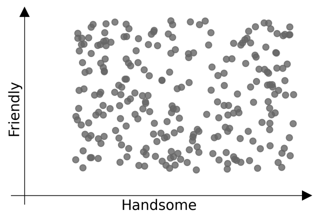
The result of the proposition can be illustrated by the following examples that are presented in [99]. The examples show the most elementary causal structures: the *chain*, the *fork*, and the *collider*.

Example 3 (Illustration of Markov Property)**.** A *chain* is a graphical structure of the form $X \rightarrow U \rightarrow Y$ as in Figure 3.5a. As an illustrative example consider the scenario where the binary variable X represents the occurrence of *fire*, U indicates whether there is *smoke* (also binary), and Y represents the activation of a *fire alarm* (also binary) [99]. The graphical structure $X \rightarrow U \rightarrow Y$ clearly describes the causal relation between these variables: Fire induces smoke with high likelihood and smoke (which is the mediator) very likely activates the fire alarm. Note that this chain is not deterministic. Occasionally, the alarm may sound without a fire present (e.g., due to a malfunction in the fire alert system), and similarly, there might be a fire without smoke (e.g., when windows are left open). We can easily conclude that in most cases fire induces smoke which triggers the fire alarm with high likelihood. Therefore, we conclude that $X \not\perp Y$. The path is in this case open and information flows from X to Y via smoke U and vice versa. However, if we condition on U, i.e. we know for instance that there is smoke, then knowing something about fire does not inform us that the fire alarm is ringing because the fire alarm is activated by the mediator smoke. In this case, the path where information flows from X to Y is blocked if we know the state of U. More formally, we have $X \perp Y \mid U$. Therefore, the statistical independence statements follow from the d-separation statements the graph $X \rightarrow U \rightarrow Y$ elicits.

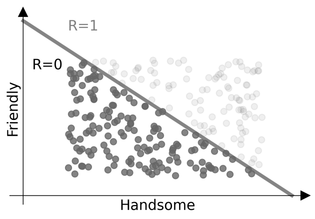
The *fork* is a graphical structure of the form $U \rightarrow X$ and $U \rightarrow Y$ as in Figure 3.5b. As an illustrative example consider the following variables: U represents the *age* of a child, X its *shoe size*, and Y its *reading skills*. We can plausibly argue that the causal structure here is a fork. Age U *causes* the shoe size X and the reading skills Y of a child. However, there is no direct causal relation between reading skills Y and shoe size X. Therefore, the variables correspond to a fork. From a statistical viewpoint, there is an obvious correlation between shoe size and reading skills: Shoe size is correlated with age and age in turn with reading skills. Therefore, the shoe size gives up information about reading skills, i.e. $X \not\perp Y$. Nevertheless, if we know the age of a child, the shoe size does not give any information about



(a) Collider structure with variables R (in relationship or not), F (friendliness) and H (handsomeness).



(b) Friendliness and handsomeness are independent Variables when pooled over all men.



(c) Friendliness and handsomeness become dependent when conditioning on R.

Figure 3.6. Adapted example "Why are handsome men such jerks?" by [100].

the reading skills and vice versa. So conditioning on age blocks the path between shoe size and reading skill and both variables become independent, i.e. $X \perp Y \mid U$. Also in this example, the statistical independence statements follow from the d-separation statements in the fork.

A more intricate example is the *collider* or v-structure. A collider has the graphical structure of the form $X \rightarrow U \leftarrow Y$ (see also Figure 3.5c). As an illustrative example consider the following variables that conform to this graph: U represents the *success* of an actress, Y describes the actress' *talent*, and X her *beauty*. Success is caused by a mix of talent and beauty. However, we can reasonably assume that beauty and talent are independent and do not exert a direct influence on each other. Therefore the underlying causality between U, X, and Y is correctly described by a collider structure. However, if we know someone is successful and we also know that she is far from beautiful, then we can conclude that she is probably very talented. Therefore, talent Y and beauty X become dependent, if we condition on success U, i.e. $X \not\perp Y \mid U$. In this example, the graphical structure contains two important d-separation statements. First, $X \perp_d Y$ which we also find in the distribution, namely X and Y are statistically independent. Second, $X \not\perp_d Y \mid U$ which allows that X can become statistically dependent on Y given U.

The fact that independent variables can become dependent if we condition on a common effect is a very important causal phenomenon and can lead to a *selection-bias* or *collider-bias* or *Berkson's paradox*:

Remark 5 (Selection or Collider-Bias)**.** *If we unknowingly condition on a common effect, then actual independent variables might become statistically dependent. This phenomenon is also known as* ***Berkson's paradox*** *[101] and has many manifestations. An example adapted from [100] illustrates very well how unknowingly conditioning on a variable might render two independent variables dependent. In this example, we consider three variables: the* ***handsomeness*** *H of a man, his* ***friendliness*** *F, and whether he is in a* ***relationship*** *or not, i.e. $R \in \{0, 1\}$. It is reasonable to assume that friendly and handsome men are more likely in a relationship since R and H are the cause for whether they end up in a relationship. We can express this via a functional relationship $R = f(F, H, N_R)$ where f is a function and N_R is some noise variable. Furthermore, we can assume that friendliness and handsomeness are independently distributed, i.e. $F \perp H$. This leads to a causal graph with a collider structure as in Figure 3.6a. For simplicity and illustrative purposes, we assume that all men who score in friendly plus handsome, i.e. $F + H$, over some threshold are in a relationship (so we assume a deterministic relation). If women are dating men,*

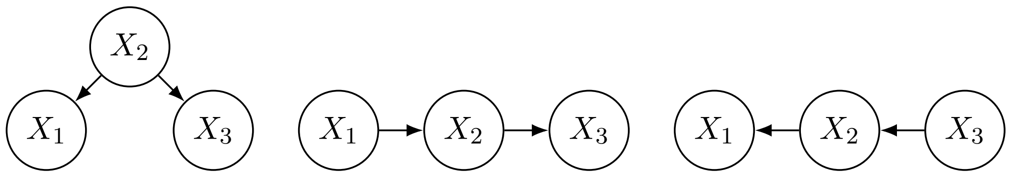


Figure 3.7. Left graph structure is also called *fork* and the other two are called *chain*. All three graphs are *Markov equivalent*, i.e. they induce the same d-separation statements.

they implicitly condition on $R = 0$, i.e. on the men who are not in a relationship. However, in this case, F and R become dependent, i.e. $F \not\perp H \mid R = 0$ as Figure 3.6b and Figure 3.6c shows. In this case, very handsome men tend to be very unfriendly, and very friendly men tend to score low in handsomeness. Therefore, the example is titled "Why are handsome men such jerks?".

The example in the remark demonstrates a case where *Reichenbach's principle* is violated: When we condition on $R = 0$ two variables become statistically dependent. This statistical relation cannot be explained by F causing H, H causing F, or a common cause of both as postulated in Reichenbach's principle. The collider examples in Remark 5 and Example 3 show also how collider structures can be used to learn about causality from data: If two variables are independent, but become dependent when conditioned on a third variable, then we might have identified a collider structure. We discuss the role of colliders for causal discovery in Section 3.9.

Markov Equivalence

We discussed that the graph and distribution obtained from an SCM satisfy a specific relationship: d-separation statements translate to statistical independence statements (see Proposition 3). We now turn our attention to characterizing graphs based on the contained d-separation statements. First of all, the following example shows that three elementary graphs can encode the same d-separation statements.

Example 4. Consider the three graphs in Figure 3.7. All three graphs encode only the d-separation statement $X_1 \perp_d X_3 \mid X_2$. Consequentially all three graphs imply the same conditional independence statement for all distributions $P_{\mathbf{X}}$ that are Markovian with respect to these graphs.

We can formally define the class of graphs that elicit the same d-separation statements. The following definition is due to [32, Definition 6.24]:

Definition 9 (Markov Equivalence of Graphs). *We denote by $\mathcal{M}(G)$ the set of distributions that are Markovian with respect to G*

$$\mathcal{M}(G) := \{P \mid P \textit{ is a distribution and Markovian w.r.t. } G\} \tag{3.23}$$

*Two graphs G_1 and G_2 are then said to be **Markov equivalent** if $\mathcal{M}(G_1) = \mathcal{M}(G_2)$. Similarly, we can define the **Markov equivalence class** of a DAG G as the set of DAGs that are Markov equivalent to G.*

It is important to note that two graphs are Markov equivalent if and only if they imply the same d-separation statements. Therefore, the graphs in Figure 3.7 are Markov equivalent. In fact, we can “graphically” validate if two graphs elicit the same set of d-separation statements [102]:

Proposition 4. *Two DAGS G_1 and G_2 are Markov equivalent if and only if they have the same skeleton and the same v-structures.*

Hence, we can conclude that the graph $X_1 \rightarrow X_2 \leftarrow X_3$ is not Markov equivalent to the graphs (fork and chains) in Figure 3.7. In Section 3.9 we will discuss the importance of Markov equivalence for causal discovery (i.e. the process of uncovering the causal structure from data).

Markov Blanket

In the following, we define the *Markov blanket* which is the smallest set of nodes that allows us to separate a target node Y from the rest of the graph.

Definition 10 (Markov blanket)**.** *Let $G = (V, A)$ be a DAG and $Y \in V$ a target node. The **Markov blanket** of Y is the smallest set $\mathbf{M} \subset V$ such that*

$$Y \perp_d V \setminus (\{Y\} \cup \mathbf{M}) \,|\, \mathbf{M} \tag{3.24}$$

If the distribution $P_{\mathbf{X}}$ is Markovian with respect to G, then the graphical separation statements transfer to the statistical conditional independence statements:

$$Y \perp V \setminus (\{Y\} \cup \mathbf{M}) \,|\, \mathbf{M} \tag{3.25}$$

In this case, the *Markov blanket* of a target variable Y is the only set of nodes necessary to predict Y. One might expect that to predict Y, we only need to include the variables from the Markov blanket. However, additional variables might help in the prediction task. For instance, the optimal predictive function estimating Y from the Markov blanket might not be encompassed within the function class considered during optimization. Hence, additional variables could aid in the prediction task. While we did not say anything about the graphical relation of the Markov blanket with the target Y in Definition 10, it is possible to fully describe the Markov blanket:

Proposition 5. *Let $G = (V, A)$ be a DAG and Y a target node. The **Markov blanket** of Y is the set of parents, children, and parents of its children:*

$$\mathbf{M} = pa(Y) \cup ch(Y) \cup pa(ch(y)) \tag{3.26}$$

The proof for this Proposition can be found in [91, Corollary 6].

Faithfulness

The Markov property states that d-separation statements translate to conditional independence statements. The *faithfulness* criterion works in the other direction, specifying how conditional independence statements translate to graphical d-separation statements. As in [32, Definition 6.33] we define

Definition 11 (Faithfulness)**.** *Let G be a DAG and $P_{\mathbf{X}}$ a distribution. Then $P_{\mathbf{X}}$ is* ***faithful*** *to G if*

$$\mathbf{A} \perp \mathbf{B} \,|\, \mathbf{C} \Rightarrow \mathbf{A} \perp_d \mathbf{B} \,|\, \mathbf{C} \tag{3.27}$$

for all disjoint node sets $\mathbf{A}, \mathbf{B}, \mathbf{C} \subset \mathbf{X}$.

While a distribution that arises from an SCM always satisfies the Markov property with respect to the corresponding graph (see Proposition 3), the same does not necessarily hold true for the faithfulness condition. In the following example adapted from [32, Example 3.34] we define a distribution via an SCM that is not faithful to the underlying graph.

Example 5. Consider the following linear Gaussian SCM that corresponds to the graph G in Figure 3.8

$$X := N_X, \tag{3.28}$$
$$Y := aX + N_Y, \tag{3.29}$$
$$Z := bY + cX + N_Z, \tag{3.30}$$

where $N_X \sim \mathcal{N}(0, \sigma_X^2), N_Y \sim \mathcal{N}(0, \sigma_Y^2)$ and $N_Z \sim \mathcal{N}(0, \sigma_Z^2)$. If $a \cdot b + c = 0$, we obtain

$$\begin{aligned}
\text{Cov}(X, Z) &= \text{Cov}(X, bY + cX + N_Z) && (3.31)\\
&= b\,\text{Cov}(X, Y) + c\,\text{Cov}(X, X) + \text{Cov}(X, N_Z) && (3.32)\\
&= b\,\text{Cov}(X, aX + N_Y) + c\,\text{Cov}(X, X) + 0 && (3.33)\\
&= ab\,\text{Cov}(X, X) + c\sigma_X^2 && (3.34)\\
&= \sigma_X^2(ab + c) = 0 && (3.35)
\end{aligned}$$

We first note that normal distributions are closed under linear combinations (i.e. a linear combination of normal distributions is also normally distributed) and that two normal distributions are independent if and only if they have 0 covariance. Therefore, we can conclude that $X \perp Z$ if $ab + c = 0$. For this configuration of coefficients, we have $X \perp Z$, despite the fact that both variables are not d-separated. Consequentially, faithfulness is violated. Note that we condition here on the empty set.

In the example, we have chosen specific values for a, b, and c such that the statistical dependency between X and Z cancels out. This requires a particular coefficient configuration. One can actually show that if the coefficients are randomly drawn from positive densities in a linear SCM, configurations leading to non-faithfulness are a null set, i.e. have zero probability [103, Theorem 3.2]. Furthermore, if we assume both faithfulness and the Causal Markov condition, d-separation statements and conditional independence statements align. We discuss the significance of this case for identifying causal relations in Section 3.9.

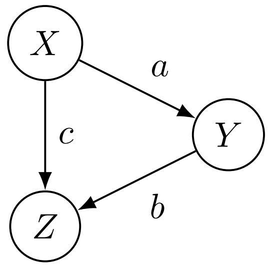


Figure 3.8. Graphical structure considered in Example 5.

3.5. Languages of Causality

The SCM that we introduced in Section 3.3 is a powerful tool for expressing causal relations and it enables causal predictions, such as interventions and counterfactuals. However, there are other representations of causality that also can express causal relations. Below, we will briefly discuss the most important ones and compare their expressive power to SCMs

Potential Outcomes The *potential outcome (PO)* framework constitutes an alternative to SCMs to ask and answer interventional and counterfactual questions. While SCMs are more popular in the computer science community, the potential outcome framework is widely used in the social sciences and statistics. Initially, the PO framework was introduced in [104] and [105] mainly in the context of Randomized Controlled Trials. [106] extended the ideas to observational data. In the following, we give a very brief overview of the framework and refer for details to [107–109].

One of the best ways to understand the PO framework is through its application of assessing a medical treatment, denoted as T. Here we adapt an example by [32, Chapter 6.9]. For instance, a patient u can either be treated ($T = 1$) or left untreated ($T = 0$). Consider the case where patient u is treated, i.e. $T = 1$. Then the outcome is deterministically described as $B_u(t = 1)$. The patient is either cured ($B_u(t = 1) = 0$) or sick ($B_u(t = 1) = 1$). What we did not observe in this case is what would have happened if the patient was not treated, i.e. $B_u(t = 0)$. This is what the counterfactual question asks for. In particular, we are interested in the difference $B_u(t = 0) - B_u(t = 1)$. This difference is also called the *unit-level causal effect*. If we have data of treated and untreated patients in addition with the outcome, then we have only access to the factual world, i.e. to one treatment condition for one patient. This is the "fundamental problem of causal inference" [110]: we can only observe either $B_u(t = 1)$ or $B_u(t = 0)$, but not both. The causal inference problem is phrased in this setting as a missing data problem and can therefore be understood as a prediction task: What is $B_u(t = 0)$ and $B_u(t = 1)$?

[110] describes two remedies to the fundamental problem of causal inference: First, one can use *scientific* insights to predict the counterfactual $B_u(t = 0)$ (assume that we already observed $B_u(t = 1)$). For instance, say patient u was treated in a previous experiment with treatment $T = 0$. If we assume that he will behave in the same way over time (*homogeneity*), then we can overcome the fundamental problem of causal inference and indeed answer the counterfactual question of what would have happened if we had not treated him, i.e. $B_u(t = 0)$. However, for this to work, we have to make the untestable homogeneity assumption. As a second approach, one could use the overall *population statistics* to achieve probabilistic statements about the causal effect. For instance, if we have a randomized controlled trial where the treatment condition is randomly assigned, then we can estimate the *average causal effect* of T:

$$\frac{1}{|U_0|} \sum_{u \in U_0} B_u(t = 1) - \frac{1}{|U_1|} \sum_{u \in U_1} B_u(t = 0) \tag{3.36}$$

where U_0 are all units that underwent treatment $T = 0$ and U_1 all units that received treatment $T = 1$. Using the overall population statistics is the way the PO framework tries to answer causal questions. For more details see [32, Section 6.9] or [110].

How does the potential outcomes framework relate to SCMs? Indeed one can show that they are both mathematically equivalent [94, Section 7.4.4]. A mathematical statement that holds in one world also holds in the other. Therefore, both frameworks are equally powerful. However, some statements might be easier to show in one world compared to the other [94, Section 7.4.4].

Graphical Models We have seen in Proposition 3 that an SCM induces a probability distribution that conforms to the causal factorization

$$P(X_1, \ldots, X_D) = \prod_{i=1}^{D} P(X_i \,|\, \mathbf{x}_{pa(i)}) \tag{3.37}$$

The underlying graph G is defined via the SCM and the factorization property is derived from the Markov property. If we abstract away the functional relations between cause-effect variables that define an SCM, we get a *graphical model*. A graphical model is described via the graph G and the causal factors in Equation 3.37. Without the causal interpretation, this is also known as a *Bayesian network* [111].

If a graphical model conforms to the underlying causality (i.e. parents in the graph correspond to the real direct causes), then we are in the position to predict interventions. For instance, if we modify mechanism $P(X_k \,|\, \mathbf{x}_{pa(k)})$ and replace it with $\overline{P}(X_k \,|\, \mathbf{x}_{pa(k)})$, then the new distribution is

$$\overline{P}(X_1, \ldots, X_D) = \overline{P}(X_k \,|\, \mathbf{x}_{pa(k)}) \prod_{i=1, i \neq k}^{D} P(X_i \,|\, \mathbf{x}_{pa(i)}) \tag{3.38}$$

In Section 3.6 we give interventions in the setting of graphical models a more thorough discussion.

While SCMs allow us to predict counterfactual questions by conditioning on the noise distribution and then use the updated SCM to predict an intervention, we cannot do something similar in graphical models rendering them slightly less powerful.

Differential Equations Coupled differential equations are a way to describe how a system evolves over time. If this description truly represents the underlying physicality, then the equations give insights into the mechanisms of the causal system [89, 97]. In this case, we can answer interventional and counterfactual questions.

Consider the following generic coupled differential equation

$$\frac{d\mathbf{x}}{dt} = f(\mathbf{x}) \tag{3.39}$$

with initial value $x(t_0) = x_0$. We assume that this coupled differential equation correctly describes the physical mechanisms of a system. Similarly as in [89] we can argue that from a given state $\mathbf{x}$ we can predict its immediate future: If f is Lipschitz, then Picard–Lindelöf theorem guarantees that, at least locally, there exists a unique solution $\mathbf{x}(t)$.

Inserting an infinitesimal dt and $d\mathbf{x} = \mathbf{x}(t + dt) - \mathbf{x}(t)$ in Equation 3.39 gives

$$\mathbf{x}(t + dt) = \mathbf{x}(t) + dt \cdot f(\mathbf{x}(t)) \tag{3.40}$$

As in [89] argued, we can read off which future values in $\mathbf{x}(t + dt)$ are caused by which present values $\mathbf{x}(t)$ from Equation 3.40. Therefore, we can directly read off the causal structure of the underlying system.

While coupled differential equations describe the system over time, SCMs can be used in a setting where the time component is ignored. Hence, in general, coupled differential equations do provide a finer physical understanding of the underlying causality compared to SCMs. [97] discuss differences between the *langugages of causality* in more depth. Table 3.1 gives an overview of the capabilities of the different languages of causality.

Representation \ **Capability**	ID prediction	interv. prediction	counterf. prediction	physical insight	learn from Data
Differential equations	✓	✓	✓	✓	÷
SCMs / PO framework	✓	✓	✓	÷	÷
Graphical models	✓	✓	✗	÷	÷
Statistical models	✓	✗	✗	✗	✓

Table 3.1. Comparison of different languages of causality as well as their capabilities. The table is adapted from [89]. ✓means satisfied, ✗means not satisfied and $\div$ means satisfied only with assumptions or under certain conditions.

In the context of robustness, we require models to be learnable from data, capable of ID prediction, and prediction under interventions. Statistical models are insufficient for these tasks. All other proposed models can handle interventions, but require additional assumptions for learnability. We hypothesize that graphical models and SCMs are amongst the most easily learnable in most relevant scenarios.

3.6. Principle of Independent Causal Mechanisms

We thoroughly discussed representations for causality and how statistical relations arise from causal ones. But why should we care? Why should we care about the exact causal relations between variables? The motivation for causal models could be phrased in one term: *robustness*. Causal models not only describe data as statistical models do, but they also allow us to predict how the data will behave if the underlying system is modified. Causal models allow us to answer the question of which "existing" knowledge is still applicable if things change. The *principle of independent causal mechanisms* gives a foundational explanation on when and why "existing" knowledge is re-usable. In Chapter 5 we operationalize this fundamental principle in order to learn knowledge that remains invariant and can be re-used if certain conditions change. As formulated in [32, Chapter 2.1], the Principle of Independent Causal Mechanism (ICM) states

Principle 2 (Indepentend Causal Mechanisms)**.** *The causal generative process of a system's variables is composed of autonomous modules that do not inform or influence each other.*

In the probabilistic case, this means that the conditional distribution of each variable given its causes (i.e. its mechanism) does not inform or influence the other mechanisms.

The principle of ICMs has two main components, an *interventional* and an *informational*. For the discussion of these two components, we consider the simplest causal scenario of two variables where one variable causes the other. In this case, the principle of ICM

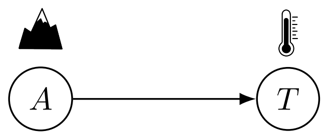


Figure 3.9. Causal graph where altitude A causes temperature T.

is termed *independence of cause and mechanism (ICM)* [112, 113]. For illustrative purposes, we elaborate on an example from [32, Chapter 2.1] consisting of two variables A and T where A represents the *altitude* of a temperature measurement device and T the *temperature* measured (see Figure 3.9). The causal direction, in this case, is that A causes T and not vice versa. A Gedankenexperiment makes this obvious. If we intervene on the altitude A by putting the measurement devices to different spots, e.g. taking them from places in the valley to the top of the mountain, we change the temperature that is measured. In contrast, if we just heat the measurement devices, we do not change their altitudes.

Intervention The principle of ICM states that the causal generative process consists of *autonomous modules* that do not *influence* each other. This has implications if we intervene/modify the observed system. In our working example, the causal generative process is described via the causal graph in Figure 3.9. We can for instance describe the causal generative process that gives rise to the data via a *Structural Causal Model (SCM)* that we defined formally in Definition 6 or a *graphical causal model* as in Section 3.5.

In an SCM, the causal generative process for our *altitute-temperature example* is described via

$$A = N_1 \tag{3.41}$$
$$T = f(A, N_2) \tag{3.42}$$

where N_1, N_2 is unobserved and follows a suitable noise distribution. f is a function that properly describes the underlying causality *and* how T arises from A and some unobserved noise N_2. The SCM allows us to change the temperature T without affecting the altitude A, i.e. we can intervene on T. This corresponds to our common knowledge about reality: we can heat the device without changing its altitude. Similarly, we can change the altitude A without changing the physical mechanism (described via $f(A, N_2)$) that relates altitude A with temperature T. Therefore, we can say that the true SCM satisfies indeed the principle of ICM – we can change one causal module without affecting the others. Note that we can also express the data via an SCM that corresponds to the graph $T \to A$ (see for instance Proposition 6). Since this SCM would not describe the underlying causality, it would not satisfy the principle of ICM.

Likewise, in a probabilistic causal model, the joint distribution between temperature T and altitude A is described via

$$P(A, T) = P(T \mid A)P(A) \tag{3.43}$$

We can apply similar considerations as with the SCM to the probabilistic causal model. For instance, if we assume that the measurement devices are all positioned in Austria a, then we obtain a distribution $P(A^a, T^a) = P(T^a \mid A^a)P(A^a)$ where the superscript indicates the country. We could also change the distribution by placing the measurement devices to Switzerland s – for instance, if the Swiss government decides to buy them. In this case, we get a new distribution, namely $P(A^s, T^s) = P(T^s \mid A^s)P(A^s)$. From a causal perspective, we have just modified the altitude A, in particular the distribution $P(A^a)$ has been changed to $P(A^s)$. The principle of ICM states that the autonomous modules which are $P(A)$ and $P(T \mid A)$ do not influence each other. This implies that we can change the altitude without affecting the underlying physical mechanism that relates altitude with temperature, i.e. $P(T \mid A)$. Therefore, we can conclude that $P(T^a \mid A^a) = P(T^s \mid A^s)$. The factorization $P(A, T) = P(T \mid A)P(A)$ is therefore special, because if we intervene on the system, e.g. by changing the altitude distribution $P(A)$ we might re-use the *causal mechanism* $P(T \mid A)$. The chain rule of probability also allows us to factorize the joint distribution into non-causal factors, i.e. $P(A, T) = P(A \mid T)P(A)$. In the following, we explain that the non-causal factorization does not necessarily satisfy the autonomy of modules. We easily obtain

$$P(A^s \mid T^s) = P(T^s \mid A^s)\frac{P(A^s)}{P(T^s)} = P(T^a \mid A^a)\frac{P(A^s)}{P(T^s)} \tag{3.44}$$

and

$$P(A^a \mid T^a) = P(T^a \mid H^a)\frac{P(A^a)}{P(T^a)} = P(T^s \mid A^s)\frac{P(A^a)}{P(T^a)} \tag{3.45}$$

So we can conclude that $P(A^s \mid T^s) = P(A^s \mid T^s)$ holds if and only if

$$\frac{P(A^a)}{P(T^a)} = \frac{P(A^s)}{P(T^s)} \tag{3.46}$$

which would require some specific entanglement between $P(A)$ and $P(T)$. This shows also that the principle of ICM can be used to discover the underlying causal relation between A and T. For instance, if we observe that by changing $P(A)$ only $P(T \mid A)$ stays invariant whereas $P(T)$ as well as $P(A \mid T)$ changes, we might prefer the causal explanation A causes T over T causes A.

A similar line of reasoning applies to a causal system consisting of multiple variables. In this case, we can change one causal factor without affecting the others. One could also say that this is the property of interventions being local. This means for an SCM that when we intervene on one variable X_i, we modify the structural assignment $X_i := f_i(\mathbf{x}_{pa(i)}, N_i)$ without changing the other structural assignments. In a graphical model, it means, that we change one of the causal factors $P(X_i \mid \mathbf{x}_{pa(i)})$ without affecting the others $P(X_j \mid \mathbf{x}_{pa(j)})$ $(i \neq j)$. In both cases, we can extensively reuse the other, unmodified components. This is not necessarily the case for data-generating processes that are non-causal. For instance, the data distribution $P(X_1, \dots, X_D)$ can be factorized along any index ordering, i.e.

$$P(X_1, \dots, X_D) = \prod_{i=1}^{D} P(X_{\pi(i)} \mid X_{\pi(i-1)}, \dots, X_{\pi(1)}) \tag{3.47}$$

where $\pi\colon \{1, \dots, D\} \to \{1, \dots, D\}$ is an arbitrary permutation, i.e. bijective function. As in the two variable case, we might expect for the non-causal factorization all (or most) factors to change, even if we just intervened on one causal factor.

It is important to mention that the principle of ICM only holds under certain conditions in our example. For instance, if we would place the measurement devices from places with continental climate to places with maritime climate the mechanism $p(T \mid A)$ might change as well.

Information The principle of ICM has also an informational component that states the causal *autonomous modules do not inform each other*. So knowing $P(X_i \mid \mathbf{X}_{pa(i)})$ does not give information about the other causal mechanisms $P(X_j \mid \mathbf{X}_{pa(j)})$. This is much more difficult to formalize and exploit. [114] propose to use the Kolmogorov Complexity to formalize this property of the principle of ICMs. Two mechanisms $p(X_i \mid \mathbf{X}_{pa(i)})$ and $p(X_j \mid \mathbf{X}_{pa(j)})$ share information if mechanisms can be described by a shorter program if we also know the program of the second mechanism. For more details on the informational component see for instance [32, Chapter 4.1.0] or [32, Chapter 6.10]

In this work, we only consider the interventional component of the principle of ICM. We operationalize this principle and make it amenable to gradient-based optimization in Chapter 5.

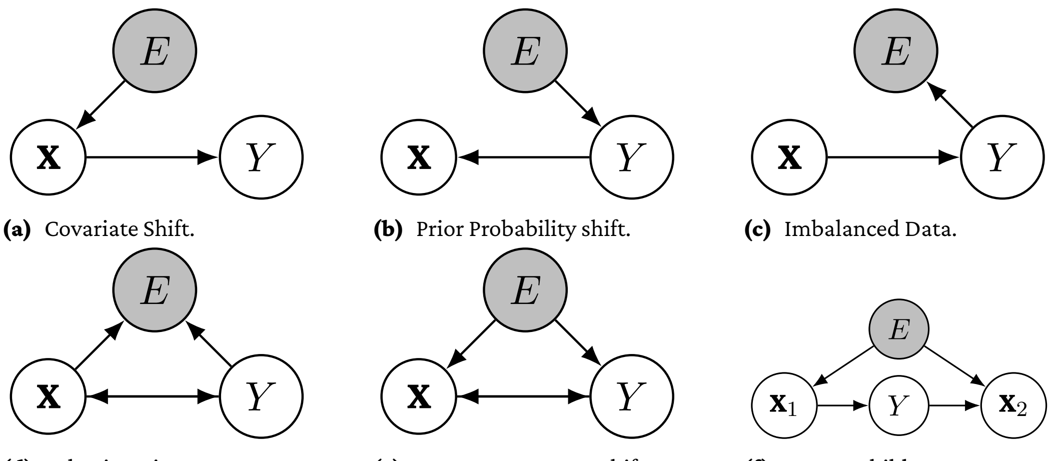
3.7. How Causal Models Can Create the Data

If we describe the data via a causal model (here an SCM), we assume some underlying causal relations in the variables. From a helicopter perspective, the data might be described via three variables: the observed data $\mathbf{X}$, the target variable Y, and an unobserved, latent variable Z that produces and relates $\mathbf{X}$ and Y.

If we consider only these three variables as entities, we can distinguish between the three scenarios described in *Reichenbach's common cause principle*. Firstly, the inputs $\mathbf{X}$ might cause Y. This might be the case when for instance someone assigned the label/value Y from $\mathbf{X}$ as it is often the case. Then the process can be described via the causal model $\mathbf{X} \to Y$. Secondly, the label Y is causing the inputs $\mathbf{X}$. This scenario might occur if the label Y is causing the observations $\mathbf{X}$. This is the case if for instance a disease referred to as $Y_{\text{ground truth}}$ is causing the observation of an MRT image X. In this case, it is important how we actually gained the label $Y = Y_{\text{measured}}$. If it is labeled by some ground truth method (e.g., an autopsy), then we can assume that $Y_{\text{measured}} = Y_{\text{ground truth}}$ is causing X, i.e. $Y \to X$. However, if X was labeled by a doctor, we can assume that $Y_{\text{ground truth}} \to X \to Y = Y_{\text{measured}}$ and $Y_{\text{measured}} \neq Y_{\text{ground truth}}$. Therefore, the same type of data can have different causal explanations depending on how it was labeled. Thirdly, the label Y and observation $\mathbf{X}$ have a common cause Z. Examples might include a person writing numbers from their intent Z. The person generates both the numbers as well as the corresponding labels Y (see [32, Chapter 1.4] for this particular example).

There are more scenarios that can describe causal relations in $P_{\mathbf{X},Y}$. For instance, some variables in $\mathbf{X}$ are causes of Y and some are effects (see e.g. Figure 3.10f). We focused here on the simple cases and will discuss more scenarios in Section 3.8.

The important question is *does it matter?*. Does it make a difference how to describe the data? The difference between the three scenarios we described above seems subtle. Two different causal models might plausibly describe the data as for instance the relation between MRT image $\mathbf{X}$ and disease Y discussed above. However, only one model *truely* explains how the data is collected and labeled. Although this distinction might seem subtle, it has real consequences. [115] showed that whether additional data $\mathbf{X}$ without labels



(a) Covariate Shift. **(b)** Prior Probability shift. **(c)** Imbalanced Data. **(d)** Selection Bias. **(e)** Source component shift. **(f)** Parent-Child Structure.

Figure 3.10. Causal graphs underlying different types of distribution shifts. If an arrow points in both directions, it implies one of three scenarios: either one variable causes the other, or vice versa, or they share a common cause.

– termed semi-supervised learning – helps the prediction task depends on whether **X** is causing Y or vice versa. In particular, they found, that semi-supervised learning can help in the anticausal direction, i.e. $Y \to \mathbf{X}$, and not in the causal direction $\mathbf{X} \to Y$. Also in terms of robustness, it makes a difference how **X** and Y relate as we will discuss in Section 3.8.

3.8. Distribution Shifts and Their Relation to Invariances

In Chapter 2 we formalized the robustness/domain generalization problem using the formulation provided in Equation 2.3. From a causal perspective, differences between distributions in the family $\mathcal{P}$ can be described by interventions (see also [7]). Taking this perspective, we elaborate on the characteristics of various relevant distribution or dataset shifts[2]. Thereafter, we assume that the extracted features bear a strong causal relation to the target and investigate implications on various types of invariances. Finally, we consider all previous dataset shifts and deduce which type of invariance is required for each distribution shift respectively, to ensure robust predictions.

3.8.1. Distribution Shift from a Causal Perspective

In this subsection, we consider various relevant distribution shifts. We adopt the terminology and conceptual definitions for the "standard distribution shifts" and the domain shift from [8, Chapter 1]. For the other types of distribution shifts, we refer to the respective literature. The distribution shifts under consideration can be categorized into shifts occurring in the input space (refer to Figure 3.10), the latent space (refer to Figure 3.12), and those influenced by hidden confounders (refer to Figure 3.11). Due to the extensive variety of potential causes behind dataset shifts, we make no claim of being fully comprehensive.

[2]We use the terms *dataset shift* and *distribution shift* interchangeably.

Covariate Shift *Covariate shift* describes the scenario where the input distribution varies, while the mechanism that relates Y and X remains unchanged [116]. This can be described with the causal graph in Figure 3.10a. The joint distribution factorizes along the causal factors $P(\mathbf{X}, Y) = P(Y \mid \mathbf{X})P(\mathbf{X})$. In this case, only the distribution $P(\mathbf{X})$ changes across different environments while $P(Y \mid \mathbf{X})$ remains invariant. By internalizing the environment variable into our model, we can also formalize this situation as $P(\mathbf{X}, Y, E) = P(Y \mid \mathbf{X})P(\mathbf{X} \mid E)P(E)$. Many real-world dataset shifts can be categorized as covariate shifts. Consider for instance the scenario where hospital A utilizes MRT images to predict a patient's heart pressure, primarily dealing with severe cases. If we have the same MRT device at hospital B and all conditions are identical, except that patients with milder conditions are diagnosed, we observe a shift in the input distribution $P(\mathbf{X})$, while the mechanism $P(Y \mid \mathbf{X})$ remains invariant. Furthermore, covariate shift scenarios include adversarial samples [44].

From a modeling standpoint, we only need to learn $P(Y \mid \mathbf{X})$ which is independent of $P(\mathbf{X})$. However, if the model family is not able to represent $P(Y \mid \mathbf{X})$, the learned model may depend on $P(\mathbf{X})$. In such cases, one potential remedy is importance reweighting. This approach helps to correct the covariate shift by accounting for the differences in $P(\mathbf{X})$ across environments [116, 117].

Prior Probability Shift The *prior probability shift* or also termed *target shift* [76] or *class-prior change* [78] is the scenario where $P(Y)$ changes across environments while $P(\mathbf{X} \mid Y)$ remains invariant. This can be described as the data generating process where Y causes $\mathbf{X}$ and E causally affects Y as is illustrated in Figure 3.10b.

For instance, a dataset that has been used for prior probability shift analysis is the Autism Spectral Disorder (ASD) dataset that contains Electroencephalography (EEG) signals from children that get treated for ASD [118, 119]. The target variable Y is their treatment stage, which is either before treatment, six months post-treatment, or twelve months post-treatment [119]. In this case, the causal structure as in Figure 3.10b is plausible. Treatment Y is the cause of the EEG signal $\mathbf{X}$. Different treatment protocols, i.e. shifts in $P(Y)$ (e.g., due to different locations/datasets E), do not directly influence $\mathbf{X}$, but only via Y. We do not delve deeper into the multiple approaches proposed to address prior probability shifts [see e.g. 78, 120, 121]. It is worth noting that the graphical structure underlying the prior probability shift is Markov equivalent to that of the imbalanced data scenario.

Imbalanced Data *Imbalanced data* describes a scenario where the proportion of target labels or occurrence of target values varies with the environment, but the process $P(\mathbf{X} \mid Y)$ that relates $\mathbf{X}$ and Y remains unchanged. This scenario is depicted in Figure 3.10c and can occur for several reasons. For example, the selection of samples might vary across different environments. In fraud detection or spam detection, one class (i.e. the class fraud) is commonly underrepresented. To counter this, the training dataset is often balanced out by considering only a subset of the training data to achieve equal class proportions. However, this can lead to dataset shift when applied under real conditions.

The graphical structure in Figure 3.10c reveals that knowing Y renders $\mathbf{X}$ and E independent. As a result, the conditional distribution $P(\mathbf{X} \mid Y)$ remains invariant and predictive across different environments. Hence, computing the predictive probability $P(Y \mid \mathbf{X})$ relies on estimating the invariant likelihood, $P(\mathbf{X} \mid Y)$, and understanding the behavior of the non-invariant prior, $P(Y)$, across different environments (and in the case of regression, also $P(\mathbf{X})$). For further details see Section 2.4.

Selection bias The *selection bias* occurs due to the process of how the data is selected from an overall population distribution. In contrast to the imbalanced data shift, the selection process is also influenced by $\mathbf{X}$. This scenario can be described by a causal graph as in Figure 3.10d. In formal terms, we can describe E in the underlying SCM as $E = f_E(\mathbf{X}, Y, N_E)$ where f_E denotes a *selection function* that is influenced by both $\mathbf{X}$ and Y as well as some noise N_E. An environment emerges by conditioning on the output E of the selection function. In surveys or clinical trials, participants usually decide for themselves whether to take part. This implicitly induces a bias that is characterized by $\mathbf{X}$ and Y. Consider a medical scenario where the task is to predict the outcome Y of a specific treatment for ill patients with characteristics $\mathbf{X}$. In a study, the collected dataset might predominantly consist of mild cases among participants aged 20 to 30. This could arise from the specifics of the recruitment process or self-selection by participants. If a model is trained to predict Y from $\mathbf{X}$ on the study's data, it might experience a distribution shift when applied in a setting where patients follow different age and illness intensity distributions. The selection bias not only appears in surveys and trials but in many other scenarios as well (see for instance [8, Chapter 1.6]).

Source Component Shift *Source component shift* refers to the scenario where the data comes from a number of sources (or environments) each with different characteristics [8, Chapter 1.9]. This can be described via a graph as in Figure 3.10e where the environment directly affects the input $\mathbf{X}$ and target Y. Consider for instance the introductory example where we like to predict housing prices Y from different inputs $\mathbf{X}$ such as proximity to the next school or room sizes. In this case, the geographic location might directly influence the distribution of $\mathbf{X}$ as well as the relation between $\mathbf{X}$ and Y. This could result in a distribution shift if an "unknown" region is introduced to the model. For many more examples of this shift see [8, Chapter 1.9].

Note that the source component shift has the same underlying causal graph as in *Simpson's paradox* [32, Chapter 6.6]. The upcoming example illustrates a case of Simpson's paradox. Consider the comparison between COVID-19 case fatality rates (CFRs) as in [122]: they observed that CFRs were overall higher in Italy compared to China. However, if they consider different age groups separately the effect is reversed. CFRs are lower in each age group in Italy compared to China. This phenomenon can be explained due to the different demography of both countries. China has a younger population and older people are more affected by COVID-19. Also, the health care in Italy is better than in China. In this scenario, E represents either Chinar or Italy, while $\mathbf{X}$ denotes age and Y signifies a binary variable indicating whether a person has died due to COVID-19. The environment variable evidently impacts the relation between X and Y (health care system is better in Italy) as well as the distribution of X (different demography).

Parent-Child Structure The *parent-child structure* is a structure where the target variable Y is caused by some input variables $\mathbf{X}_1$ and in turn causes some other input variables $\mathbf{X}_2$. This might occur if for instance a target variable Y is a gene expression and inputs are many other gene expressions that can either cause or be a cause of Y [29]. We detail this structure in Chapter 5 and give a more illustrative example in Example 6.

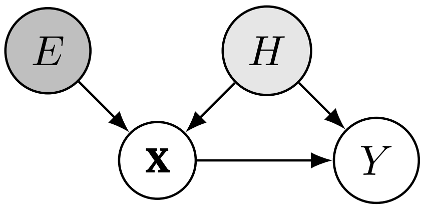


Figure 3.11. Causal model with hidden confounder H. Here, the environment variable E directly affects **X** and can also be considered an Instrumental Variable (IV).

Hidden Confounders and Instrumental Variables Hidden confounders can obfuscate the "pure" effect that X has on Y and lead to unexpected behavior under distribution shift. In the following we adapt and extend an example by [32, Chapter 9.3] to the robustness problem to show how hidden confounders can complicate the robust prediction task. The example also illustrates how access to different environments can address the robustness challenge. For technical details not included here, please refer to Section A.1. Consider the following SCM

$$X := bE + cH + N_X \tag{3.48}$$

$$Y := aX + dH + N_Y \tag{3.49}$$

where N_X, N_Y and $H = N_H$ is jointly independent and unobserved noise. We assume that $b, c, d \neq 0$ and that $\mathrm{Var}(H) \neq 0$. The environment variable E is here observed in the sense that we always know from which environment the data originates. This SCM corresponds to the graph in Figure 3.11. The hidden confounder H is not observed and affects both X and Y. The "pure" effect from X on Y would be the coefficient a and is sometimes called *average causal effect*. If we just regress on X to predict Y, we obtain an estimate for a that converges in the infinite data regime to (details in Section A.1)

$$\widetilde{a} = a + \frac{d \cdot c \cdot \mathrm{Var}(H)}{\mathrm{Var}(X)} \tag{3.50}$$

This is obviously a biased estimate. For simplicity, we assume in the following that all variables have zero expectancy. If we use this biased estimate for prediction, we obtain (cf. Section A.1)

$$\mathbb{E}_{X,Y}[(\widetilde{a}X - Y)^2] < \mathbb{E}[(dH + N_Y)^2] \tag{3.51}$$

A model based on $\widetilde{a}$ achieves a smaller predictive loss in Equation 3.51 compared to the one attained by the causal model (see Section A.1 for details):

$$\mathbb{E}[(aX - Y)^2] = \mathbb{E}[(dH + N_Y)^2] = \mathbb{E}[d^2H^2] + \mathbb{E}[N_Y^2] \tag{3.52}$$

Conclusively, we would prefer model $\widetilde{f}(X) = \widetilde{a}X$ over $f(X) = aX$ in terms of predictive loss. However, $\widetilde{f}$ is not robust to environment changes. Consider the case where the environment $E = 0$ eradicates the effect of H on X, i.e. $X := N_X$. In this case, the biased model $\widetilde{f}$ introduces an unnecessary bias (see Section A.1 for details):

$$\mathbb{E}[(\widetilde{a}X - Y)^2] = \mathbb{E}[(\widetilde{a} - a)^2X^2] + \mathbb{E}[(dH + N_Y)^2] \tag{3.53}$$

The causal model that only uses the "pure" causal effect from X to predict Y is robust with respect to environment changes and attains a smaller predictive loss:

$$\mathbb{E}[(aX - Y)^2] = \mathbb{E}[(dH + N_Y)^2] \tag{3.54}$$

There are many real-world examples where a confounding variable is unobserved. For instance, [123] aims to estimate the effect of *smoking during pregnancy* (variable X) on the *birth weight* of the born baby (variable Y) from only observational data. The hidden confounder H comprises unobserved *maternal characteristics* that can influence both X and Y. We extend this example below and consider an interesting environment variable E. Note that an intervention in terms of strictly forbidding a pregnant woman to smoke would eradicate the effect of H on X similar to our example above. In this case, a regression model relying on $\widetilde{a}$ might deliver inferior results compared to the causal model based on a.

So how could we estimate the "pure" effect of X on Y? One way to do this is to use *instrumental variables* (IVs). An instrumental variable (IV) is a variable that is (i) independent of the hidden confounder H, (ii) dependent on X and (iii) affects Y only through X [32, Chapter 9.3]. An instrumental variable can also be interpreted as an environment variable. We show a graph that corresponds to the requirements of an IV in Figure 3.11. Since E is independent of H and N_X, we can consider $cH + N_X$ as noise

$$X := bE + (cH + N_X) \tag{3.55}$$

Hence, we can consistently estimate b without introducing any biases. The target variable then becomes

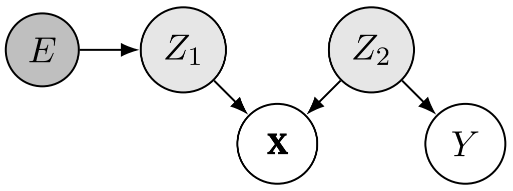
$$Y := aX + dH + N_Y = a(bE) + [a(cH + N_X) + dH + N_Y] \tag{3.56}$$

since bE is independent of the noise $a(cH+N_X)+dH+N_Y$, we can consistently estimate a by regressing on bE. This two-stage procedure is also called *two-stage least squares.*

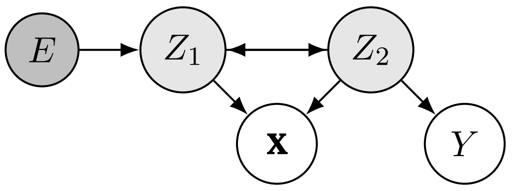
In the example above of estimating the "pure" causal effect of *smoking during pregnancy* on *birth weight of the born baby*, [123] use cigarette taxes that might vary across states and times. Cigarette taxes are independent of the maternal characteristics (condition (i)), they have an effect on the smoking behavior of people including pregnant women (condition (ii)) *and* they have no direct effect on the birth weight and affect the birth weight only through the smoking behavior (condition (iii)).

The environment variable or IV is here used to uncover the "pure" causal effect represented via the coefficient a. The Gedankenexperiment presented above shows that intervening in the system (e.g. by eradicating the effect of the hidden confounder) renders the model based on a more robust.

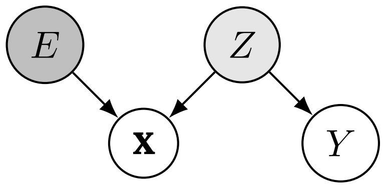
Appearance Shift What we refer to as *appearance shift* was proposed in [124] as a type of dataset shift. It is characterized by a latent model with a latent space comprising two sub-spaces. One sub-space is *environment-specific*, i.e. it changes with environmental differences, while the other sub-space is *environment-invariant* and remains invariant across distinct environments. The causal graph representing appearance shift is depicted in Figure 3.12a. Image classification tasks often exhibit an appearance shift where an environment-specific variable Z_1, such as style or image appearance varies across environments (e.g. cartoons, paintings, art). In contrast, there is often an environment-invariant *content* element,



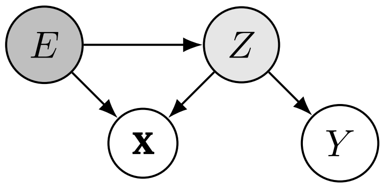
(a) Appearance shift



(b) Appearance shift (variant II)



(c) Domain shift



(d) Domain shift (variant II)

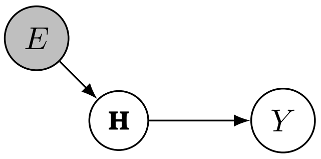
Figure 3.12. Causal graphs depicting the underlying types of distribution shifts, specifically in the context of latent variable models. If an arrow points in both directions, it implies one of three scenarios: either one variable causes the other, or vice versa, or they share a common cause.

like the consistent shape of an animal for animal classification tasks. A robust classifier would rely on the invariant signals (content) within the image. We can easily come up with interesting variants of the appearance shift where, for instance, the environment-specific and the environment-invariant information are dependent, as illustrated in Figure 3.12b and exemplified in the NICO dataset [125].

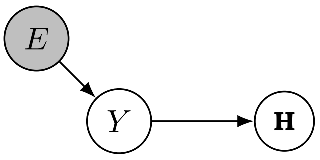
Domain shift In some cases, we do not have an actual distribution shift of the significant variables, but of how they show themselves. This can be described via a latent variable model as in Figure 3.12c where Z represents the crucial signal and $\mathbf{X}$ its appearance which varies across environments. Only through Z flows information from $\mathbf{X}$ to Y. The goal therefore is to recover Z that is not affected by the environment in order to predict Y reliably. This scenario is also called *domain shift* [8, Chapter 1.8]. Important examples occur when the measurement devices switch across environments or when numbers change their meaning (e.g., in inflation). We can also define a variant of the domain shift where E affects Z directly (see Figure 3.12d). This scenario could occur when the distribution of Z varies across environments. For instance, different hospitals (which are environments here) not only use distinct measurement devices to collect data, but the distribution of patients itself differs across hospitals.

3.8.2. Invariances from a Causal Perspective

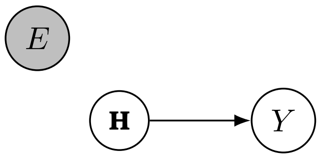
Now, we interpret selected features $\mathbf{H} = h(\mathbf{X})$ as either causing or being caused by the target variable Y. It offers valuable insights into the elicited kind of invariance, to causally relate $h(\mathbf{X})$ with Y. For instance, if we consider an image classification task where the *content* in the image $\mathbf{X}$ is the cause of the *label* Y. Assuming that $h(\mathbf{X})$ captures only the content, we could conclude that $h(\mathbf{X})$ causes Y. A model that predicts Y from $h(\mathbf{X})$ is then a robust model that does not get distracted from appearance/style which might deviate from environment to environment (see also appearance shift above). This model satisfies the causal invariance as we explain below. For an overview of our results see Figure 3.13.



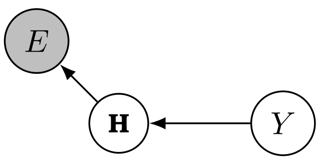
(a) Causal Invariance: ✓
Anti-Causal Invariance: ✗
Feature Invariance: ✗



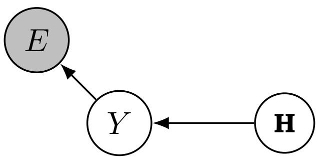
(b) Causal Invariance: ✗
Anti-Causal Invariance: ✓
Feature Invariance: ✗



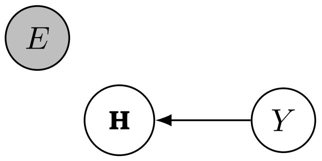
(c) Causal Invariance: ✓
Anti-Causal Invariance: ✓
Feature Invariance: ✓



(d) Causal Invariance: ✓
Anti-Causal Invariance: ✗
Feature Invariance: ✗



(e) Causal Invariance: ✗
Anti-Causal Invariance: ✓
Feature Invariance: ✗



(f) Causal Invariance: ✓
Anti-Causal Invariance: ✓
Feature Invariance: ✓

Figure 3.13. Different scenarios where a type of invariance is satisfied. Here $\mathbf{H} = h(\mathbf{X})$ is an extracted feature and Y is the target variable. We assume that Y is either causing or being caused by $\mathbf{H}$.

For our considerations, we assume the distribution satisfies the *Markov property* and *faithfulness* with respect to the respective graph. This implies that d-separation and conditional independence statements coincide. We illustrate various scenarios in Figure 3.13, allowing us to easily verify which form of invariance holds:

- $Y \perp_d E \mid h(\mathbf{X})$ which implies $Y \perp E \mid h(\mathbf{X})$ (causal invariance)
- $h(\mathbf{X}) \perp_d E \mid Y$ which implies $h(\mathbf{X}) \perp E \mid Y$ (anti-causal invariance)
- $h(\mathbf{X}) \perp_d E$ which implies $h(\mathbf{X}) \perp_d E$ (feature invariance)

For instance, the graph $E \to h(\mathbf{X}) \to Y$ in Figure 3.13a contains the d-separation statement $E \perp_d Y \mid h(\mathbf{X})$. From the Markov property, it follows $E \perp Y \mid h(\mathbf{X})$ which conforms to the *causal invariance* as defined in Definition 1. In contrast, we have $E \not\perp_d h(\mathbf{X})$ and $E \not\perp_d h(\mathbf{X}) \mid Y$ and therefore, we obtain $E \not\perp h(\mathbf{X})$ and $E \not\perp h(\mathbf{X}) \mid Y$, excluding the feature and anti-causal invariance. Thus, this graph only satisfies the causal invariance. Similarly, we can relate different types of invariances to other scenarios as done in Figure 3.13.

If E affects both $h(\mathbf{X})$ and Y, then we cannot d-separate $h(\mathbf{X})$ from E (as required for the feature invariance), Y from E (as required for the causal invariance) or $h(\mathbf{X})$ from E (as required for the anti-causal invariance). Due to the faithfulness assumption, we can conclude that no invariance form holds in case E affects both Y and $h(\mathbf{X})$. Hence, Figure 3.13 shows all scenarios where a form of invariance is possible.

3.8.3. Invariances and Distribution Shifts

Here, we temporarily set aside the complexities associated with finding an invariant representation, focusing instead on the fundamental feasibility of discovering a representation $h(\mathbf{X})$ that might exhibit a corresponding invariance property – results can be found in Table 3.2. In the case of latent variable models such as in Figure 3.12, we assume that the feature extractor h is able to recover the latent variables. Therefore $h(\mathbf{X})$ can be either $\mathbf{X}$, a selection/combination of variables of $\mathbf{X}$, or a selection of latent variables. Arguing with the d-separation statements, we can identify which types of invariances might be helpful under which kind of dataset shift. For all dataset shifts in Figure 3.10, Figure 3.12 and Figure 3.11 we can check if

- ▶ $Y \perp_d E \mid h(\mathbf{X})$ which implies $Y \perp E \mid h(\mathbf{X})$ (causal invariance)
- ▶ $h(\mathbf{X}) \perp_d E \mid Y$ which implies $h(\mathbf{X}) \perp E \mid Y$ (anti-causal invariance)
- ▶ $h(\mathbf{X}) \perp_d E$ which implies $h(\mathbf{X}) \perp_d E$ (feature invariance)

where $h(\mathbf{X})$ is equal to $\mathbf{X}$ or some variable selection of $\mathbf{X}$. Therefore, it can easily be seen that the *selection bias* and *source component shift* do not satisfy any invariance for $h(\mathbf{X}) = \mathbf{X}$. The *children-parent case* only satisfies the causal invariance for $h(\mathbf{X}) = X_1$. The results for Figure 3.10 are summarized in Table 3.2 and can be effortlessly verified.

For the latent variable models in Figure 3.12, we can similarly check whether we can find an invariance. As an example, we consider the *appearance shift* in detail. If $h(\mathbf{X}) = Z_2$, then we obtain $h(\mathbf{X}) \perp_d E$ as well as $E \perp_d Y \mid h(\mathbf{X})$ and $E \perp_d h(\mathbf{X}) \mid Y$. Due to the Markov property, we therefore obtain that all three types of invariances hold. Conclusively, the predictive *content* information Z_2 of $\mathbf{X}$ satisfies all invariance properties. Note that one exception appears in the appearance shift. Here, we also achieve $E \perp_d Y \mid Z_1$. This implies the appearance/style variable Z_1 that varies across environments satisfies the causal invariance. In this case, the sole invariance property is not indicative of our goal of finding the content information Z_2. Therefore, the predictive model $P(Y \mid Z_2) = P(Y)$ remains still invariant but delivers no further information about Y compared to the prior information $P(Y)$. Therefore, Z_2 has no utility for the feature extractor h in the domain generalization formulation in Equation 2.7. The results for the other latent variable can be found in Table 3.2 and can be verified similarly via the d-separation statements.

The hidden confounder model with confounder W surprisingly does not satisfy any invariance property. Due to the collider structure $E \rightarrow \mathbf{X} \leftarrow W$, information flows from E to Y if we condition on $\mathbf{X}$, i.e. $E \not\perp Y \mid \mathbf{X}$ (no causal invariance). The feature invariance and anti-causal invariance are obviously not satisfied. All results can be found in Table 3.2.

Invariance / **Shift**	Causal Invariance	Anti-Causal Inv.	Feature Invariance
Covariate shift	✓	✗	✗
Prior shift	✗	÷	✗
Imbalanced data	✗	÷	✗
Selection Bias	✗	✗	✗
Source Component Shift	✗	✗	✗
Parent-Child Structure	✓	✗	✗
Domain Shift	✓	✗	✓
Domain Shift (var. II)	✗	✗	✗
Appearance Shift	✓	✓	✓
Appearance Shift (var. II)	✓	✓	✗
Hidden Confounder and IV	✗	✗	✗

Table 3.2. Which type of *invariance* can be achieved under different types of *dataset shift*? ✓indicates that the invariance is satisfiable, ✗indicates that the type of invariance is unsatisfiable and $\div$ indicates that the invariance is satisfiable, but for the prediction task additional information, such as $P(Y, E)$ or $P(\mathbf{X}, E)$, is required.

In Table 3.2, we outline the types of invariances achievable under different distribution shifts. However, not all distribution shifts allow for an advantage over a standard approach by considering the corresponding invariance. Specifically, if a standard approach would trivially satisfy the necessary invariance, as in the case of covariate shift. Because of the relationship $P(Y \mid \mathbf{X}, E) = P(Y \mid \mathbf{X})$, a standard model automatically satisfies the causal invariance. This might explain why current Domain Generalization (DG) methods do not consistently outperform a simple baseline on numerous benchmark datasets, as demonstrated in [14, 51, 61, 62]. For instance, well-known benchmark datasets like PACS and OfficeHome (refer to Figure 2.1) appear to undergo a covariate shift. However, many methods evaluated on these datasets are specifically designed to approximate the causal invariance (e.g., [34, 126]). This might explain why DG methods often fail to beat a simple baseline model on several datasets. This asks for further research to identify the type of distribution shift in real-world datasets.

It is also important to note that besides the difficulty of identifying and finding the right type of invariance, there exist other challenging requirements associated with distribution shift, such as extrapolation, discussed in Subsection 4.5.4.

3.9. Causal Discovery

The goal of causal discovery is to identify the *true* SCM that gives rise to the data. We do not involve ourselves here in any philosophical discussion on what a *true* SCM is or means. We consider *true* here pragmatically in the sense that interventions in the "real world" and counterfactual questions dealing with the "real world" can be answered or predicted by the *true* SCM. Put a little differently, the *true* SCM corresponds to the real world in the sense that outcomes due to modifications and interventions to the real world can be predicted by the corresponding modifications/interventions to the SCM. While a full description of an SCM encompasses many components (e.g. noise distributions, functions in the assignments), the main focus of the causal discovery task is the identification of the graph that corresponds to the SCM. This is also called structure identification. If the graph is given, the functions in the structural assignments can be estimated (e.g., via regression). While estimating the functions in the structural assignments is non-trivial, it is not the focus of most causal discovery literature.

In the following, we give some answers under which conditions the quest for causal discovery is even possible, i.e. identifiability is achievable. Afterward, we introduce two important methods that aim to find the true SCM.

3.9.1. Graph Identifiability with Interventions

Randomized Controlled Trials (RCT) are widely considered the gold standard to learn about the underlying causal relations. In an RCT, groups are randomly assigned to different treatment conditions. After the treatment outcome is measured, one can determine the causal relations (see for instance [32, Example 6.15]). However, in most cases, RCTs are not doable for various reasons. They can be expensive (e.g., in the medical sector), technologically infeasible (e.g., too many variables like in genetics), or would violate ethical considerations. Additionally, RCTs are designed for variables with finite support, but in many relevant applications variables have infinite or continuous support.

There exists a plethora of work on the topic of causal discovery when non-randomized interventions have been performed (see also Section 5.2 or [32, Chapter 7.2]). In our work in Chapter 5 we contribute to the topic of causal discovery under interventions[3].

3.9.2. Graph Identifiability from Purely Observational Data

Time is a crucial signal for uncovering causal relationships [32, Chapter 10]. At a fine-grained time scale, we can argue that a cause must precede an effect. However, the data might lack a time component due to various reasons, such as processes are too rapid to record – common in fields like cell biology – or when the time element is simply unavailable, as illustrated in the temperature-altitude example in Figure 3.9.

Here, we delve into the causal discovery task specifically in scenarios where the time component is absent. Learning causality from purely observational data (without time) seems impossible at first sight. Causal relations are asymmetric while most statistical relationships are symmetric (e.g., correlation or mutual information). Therefore, causal relations cannot be uncovered by naively applying symmetric measures. To uncover the graph that underlies the *true* SCM, we need to face the question of whether there exists more than one SCM that could elicit the same data distribution $P_{\mathbf{X}}$. It turns out that SCMs are very powerful and usually many SCMs can elicit the distribution $P_{\mathbf{X}}$. First of all, we refer to the fact that if a distribution is Markovian with respect to a graph, we can find a corresponding SCM that produces the SCM [127, Proposition 7.1.].

Proposition 6. *Let $P_{\mathbf{X}}$ be a distribution that has a density with respect to Lebesgue measure. We assume it is Markovian with respect to G. Then there exists and SCM S that corresponds to G and elicits the distribution $P\mathbf{X}$.*

In the following remark, we discuss the fact that actually many graphs G satisfy the requirement in Proposition 6.

Remark 6. *Let $P_{\mathbf{X}}$ be a distribution. Furthermore let G be a DAG that is complete, i.e. for any node there exists a direct path to any other node except itself. Since G does not contain any d-separation statements, the Markov property holds for any distribution. So, it is evident that $P_{\mathbf{X}}$ is Markovian with respect to G. Proposition 6 implies that we can start from a full DAG G and find a corresponding SCM that elicits $P_{\mathbf{X}}$.*

There exist $D! = 1 \cdot 2 \cdot \dots \cdot D$ complete graphs with D variables. Thus, we can conclude that at least $D!$ SCMs exist that elicit the distribution $P_{\mathbf{X}}$.

In Remark 6 we discussed that a huge amount of SCMs can explain a given distribution. This renders the identification task impossible, at least from a theoretical standpoint. As it turns out, we can introduce certain assumptions that might be plausible and render the identification task possible. Some assumptions lead to a one-to-one mapping between graphs and distributions and some assumptions allow us to identify a class of graphs that can explain the data distribution. Over the years many assumptions have been proposed that render the identification task possible (for an overview see e.g. [32, 128, 129]). In the following, we briefly discuss two common assumptions to achieve identifiability.

[3]In our case, interventions correspond to environments.

Faithfulness In Remark 6 we showed that the amount of SCMs that correspond to a given distribution grows at least factorial with the number of variables. The explanation is that a complete DAG contains no d-separation statements and therefore the Markov property is trivially satisfied for any complete graph. However, if we assume the distribution is faithful to a graph, (statistical) conditional independence statements translate to d-separation statements. This implies that if we can find any conditional independence statements in the data distribution, the distribution is not faithful to a complete graph. Hence, the faithfulness assumption can reduce the amount of SCMs that elicit the distribution drastically. Remarkably, the reduction is very strong, enabling us to effectively characterize the graphs that align with the given distribution.

It is easily seen that with the faithfulness assumption, a distribution $P_{\mathbf{X}}$ has a one-to-one correspondence to its *Markov equivalence class* as defined in Definition 9. In particular Proposition 4 shows that we can identify the skeleton as well as all colliders from the observed distribution under the faithfulness assumption.

We can therefore conclude that the *chain* and *fork* as considered in Example 3 are not identifiable, but the collider is identifiable. So if we find two independent variables and find out that they become dependent when we condition on a third variable, we identify a collider structure. The collider structure is instructive since it also shows that considering more variables can ease the causal discovery task. For instance, if we consider just two dependent variables, then the causal direction is not detectable under the faithfulness assumption. However, if we consider a third variable and find a collider structure, then we can identify the underlying causality.

Model Restrictions While the faithfulness assumption allows us to identify the true causal graph up to the Markov equivalence class, assuming restrictions on the structural assignments enables us to identify the one and only true causal graph. As it happens, the underlying graph G becomes identifiable if we assume the structural assignments in the SCM that produces the data are from a certain class of functions, i.e.

$$X_j := f_j(\mathbf{X}_{pa(j)}, N_j) \tag{3.57}$$

where f or N_j is restricted. For instance, SCMs produced from linear functions with non-Gaussian noise i.e.

$$X_j := \sum_{k \in pa(j)} b_{jk} X_k + N_j \tag{3.58}$$

is completely identifiable [130]. This means if we assume that the data is produced by a model like in Equation 3.58, then the mapping from graphs G to distribution $P_{\mathbf{X}}$ is a one-to-one mapping. Interestingly, even if there are just two variables, we can identify the direction if the model is produced as in Equation 3.58. While the linearity assumption in Equation 3.58 might seem very restrictive similar identifiability results have been shown for more expressive models as for instance *additive noise models* of the form $X_j := f_j(X_{pa(j)}) + N_j$ [127]. For a more extensive oversight of different model restriction assumptions that lead to identifiability see [32, Chatper 7]

3.9.3. Causal Discovery Methods

In the following, we consider two causal discovery methods in more detail. One that deals with purely observational data (independence-based methods such as the PC-algorithm) and one that also exploits information that shows across different environments (ICP method). We will compare our proposed method in Chapter 5 to these algorithms.

It is important to point out that there exist many more algorithms that aim to solve the causal discovery problem. In most cases, they rely on a different form of *causal signal* to infer causal relations. For instance, *independence-based* methods like the PC-algorithm try to exploit conditional-independence statements to uncover the true causal graph. Some methods such as the ICP method [29] or the method we propose in Chapter 5 aim to exploit *invariances* that show across environments to learn about the underlying causality. *Score-based methods* assign a score to each possible graph indicating how well the causal graph matches the data. Different search techniques have been proposed to search through all graph candidates and choose the graph with the highest score [32, Section 7.2]. This could be used for instance in combination with the restricted model assumption as discussed above. The graph that gives the best fit under a certain model class (e.g. additive noise models) is then the true causal model (see for instance [127]). For a more in-depth oversight of different causal discovery methods consider for example [32, 128, 129].

Independence-Based Methods Independence-based methods estimate a set of causal graphs from purely observational data. They proceed in two steps:

(i) Skeleton estimation

(ii) Orientation of edges

In the first step, the skeleton (i.e. the graph without directed edges) is estimated. This is possible due to the following observation

> *Two variables X, Y are directly connected if and only if there is no set of variables $\mathbf{A} \subset V \setminus \{X, Y\}$ of nodes with $X \perp Y \mid \mathbf{A}$.*

This follows directly from the faithfulness assumption: If we find a suitable set of variables $\mathbf{A} \subset V \setminus \{X, Y\}$ with $X \perp Y \mid \mathbf{A}$, then we obtain $X \perp_d Y \mid \mathbf{A}$, i.e. X and Y are not directly connected. However, going through all possible variable sets $\mathbf{A}$ and testing for $X \perp Y \mid \mathbf{A}$ is infeasible if we consider many variables. The *PC-Algorithm* considers different subsets $\mathbf{A}$ in a more elegant order [103]. While the PC-Algorithm relaxes the computational demands resulting from a naive search, conducting nonparametric conditional independence tests with a finite amount of data remains highly challenging [32, Chapter 7.2].

In the second step, we can orient some edges in the graph. Proposition 4 indicates that we can orient all colliders. A collider is a structure of the form $X \to Y \leftarrow Z$ where X and Z are not directly connected. In Remark 5 we have discussed that these structures leave a particular causal signal in the data, namely

- $X \perp Z$, and
- $X \not\perp Z \mid Y$

Therefore, we can investigate structures of the form $X - Y - Z$ and test whether they conform to a collider. Since we assumed that the causal graph is a DAG (in particular an acyclic graph), we can also orient all edges that would contradict the acyclicity assumption. Meeks' orientation rules are a set of orientation rules that have been shown to be complete [131].

ICP The goal of the *invariant causal prediction (ICP)* method due to [29] is to find the direct causes of some target variable Y. The ICP method exploits invariances that show across different environments to learn about the causal parents of Y. In particular, it uses the concept of causal invariance that we introduced in Definition 1.

We say a variable selection $\mathbf{X_A}$ satisfies the null hypothesis if the conditional distribution $P(Y \mid \mathbf{X_A})$ is invariant with respect to the environment. Put more formally, the null hypothesis $H_{0,\mathbf{A}}(\mathcal{E}_{seen})$ is true if and only if $\mathbf{X_A}$ satisfies the *causal invariance* property as defined in Definition 1 across $\mathcal{E}_{seen}$.

The estimator is then defined as

$$S(\mathcal{E}_{seen}) := \bigcap_{\mathbf{A}:\ H_{0,\mathbf{A}}(\mathcal{E}_{seen}) \text{ is true}} \mathbf{A} \tag{3.59}$$

$$\tag{3.60}$$

The basic assumption in the ICP framework is that the target variable given its causal parents does not change across environments, i.e. $H_{0,pa(Y)}$ is true. This implies in particular that the variable set $pa(Y)$ is always one term in the intersection in Equation 3.59 and consequentially

$$S(\mathcal{E}_{seen}) \subset pa(Y) \tag{3.61}$$

This also shows that $S(\mathcal{E}_{seen})$ is a conservative estimator in the following sense: it opts to include fewer or no variables at all. For instance, if we have only one seen environment, then $S(\mathcal{E}_{seen}) = \emptyset$ and therefore $S(\mathcal{E}_{seen})$ would refrain from including variables. In [29, Theorem 1] they show statistical guarantees in the sense that

$$P(S(\mathcal{E}_{seen}) \subset pa(Y)) \geq 1 - \alpha \tag{3.62}$$

if we have a hypothesis test for $H_{0,\mathbf{A}}(\mathcal{E}_{seen})$ to level α.

In [29, Theorem 2] they show also that the estimator can indeed identify the direct causes of Y, i.e. $S(\mathcal{E}_{seen}) = pa(Y)$. However, they require strong conditions for this identifiability result (e.g. linear models and strong interventions on all variables).

In Chapter 5 we propose to exploit similar causal principles to learn about causal relationships as the ICP framework does. However, we are not required to perform an extensive search as needed in Equation 3.59 and we do not restrict ourselves to linear models. We discuss differences compared to our framework in more detail in Section 5.2.

4 Deep Learning

In this chapter, we cover the basics of deep learning, including a brief introduction to machine learning, maximum likelihood estimation, essential information theoretic concepts, neural networks, and generalization. We furthermore discuss generalization in connection to robustness and introduce important models that relate to robustness or are used in this work.

4.1. A short introduction into Machine Learning

A concise definition of what machine learning is was given by Mitchell [132, Chapter 1.1]:

> *A computer program is said to learn from experience* Exp *with respect to some class of tasks* Tas *and performance measure* Per*, if its performance at tasks in* Tas*, as measured by* Per*, improves with experience*

While this definition includes most of the elements that constitute Machine Learning, it leaves one of the most important aspects implicit: generalization. By improvement, we mean improvement on new "experience" or data. This is basically the main goal of machine learning: to generalize from *known* data to *unknown* data [1]. The generalization aspect delineates Machine Learning from pure optimization. In Chapter 4.5 we take a closer look at generalization. Now, we give a brief overview of the various shapes that different *Tasks* Tas, *Performance Measures* Per, and *Experiences* Exp can take, following [38, Chapter 5.1]. Afterward, we show how different machine learning approaches can be categorized.

4.1.1. Taks, Performance Measure and Experience

Task What qualifies as a machine learning task can vary widely and usually would require some form of intelligence to be solved. We just present a selection of the most prevalent tasks and do not claim to be comprehensive in this regard.

The *classification* task asks to find a function $f\colon \mathcal{X} \to \{1, 2, \dots, C\}$ that predicts a class label given some input. The space of applications for classification is almost limitless, ranging from classifying tumors in images [9] to predicting objects (e.g., the animal class) in images [125]. Often the classification task is not perfectly solvable for several reasons, for instance, due to unobserved influences (also called noise). Unsolvability is the norm and therefore we usually aim to learn the conditional probability $P(y \mid X = x)$ where x represents the input and $y \in \{1, 2, \dots C\}$ a class. A prediction can then be performed by choosing the class that achieves the greatest likelihood (see also Equation 4.49). The advantage is that we can provide an uncertainty estimation that gives information on how trustworthy the prediction is. In *regression* the goal is not to predict a class from an input, but a numerical value, i.e. to find a function $f\colon \mathcal{X} \to \mathbb{R}$. The variety of regression tasks is nearly boundless, including applications such as predicting a poverty index due to a satellite image [133]. Also, in the regression setting, we can equip predictions with an uncertainty estimation (see e.g., [134]). Another interesting application is *data generation*. The goal here is to find a generator $G\colon \mathcal{Z} \to \mathcal{X}$ that transforms samples from a simple distribution (e.g., Gaussian noise) to samples from a more complex distribution (e.g., images of cells obtained via microscopy).

Performance Measure The goal of machine learning is to obtain a model that is successful at fulfilling its intent. Expressing an intent formally can be quite challenging and depends strongly on the application.

In classification, we usually aim to find a model that gives good predictions. It is common practice to measure "goodness" in terms of accuracy, i.e. the number of correct classifications divided by the number of all classifications. However, in imbalanced datasets the accuracy can be deceiving. Consider for instance the binary classification task on a dataset where 99% of all instances belong to class 0 and 1% to class 1. In this case, a model that simply predicts 0 achieves an accuracy of 99%. In this case, one would prefer performance measures that are more suitable for imbalanced datasets such as the F-measure (see e.g., [135] for more details). A typical performance measure in one-dimensional regression is the L2-Loss, i.e.

$$\mathbb{E}_{\mathbf{X},Y}[(Y - f(\mathbf{X}))^2] \tag{4.1}$$

where $Y \in \mathbb{R}$ is the ground-truth value and $f(\mathbf{X})$ the corresponding prediction. We give a thorough theoretical interpretation of the L2-Loss in Subsection 4.4.2.

Experience In machine learning, one commonly distinguishes between *supervised*, *unsupervised learning*, and *reinforcement learning* algorithms. These algorithms depend on the experience (or data) they process [136].

In *supervised learning*, we assume that we are given data $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ of input-output pairs $(\mathbf{x}_i, y_i)$. Depending on the data modality of the output (e.g., categorical vs. continuous) we are usually in the classification or regression setting. We denote the set $\mathcal{D}$ as the training set and n as the number of training examples. The inputs $\mathbf{x}$ can vary from low-dimensional vectors to high-dimensional complex structures such as images or graphs. Note that the output y_i can take more complex forms beyond a one-hot encoded class vector or one-dimensional value. For instance, in semantic segmentation, we aim to predict a class label for each pixel in the image. One main drawback of supervised learning is that it is usually expensive to assign an output y_i to an input $\mathbf{x}_i$. For instance, on the crowdsourcing marketplace Amazon Mechanical Turk, it costs 0.012$ to assign a class to an image [137]. The famous dataset ImageNet contains currently 14,197,122 images [138, 139] which would add up to $14,197,122 \cdot 0.012\$ = 170,365\$$ labeling costs for the whole dataset. If we were to label this dataset multiple times to reduce labeling errors, the labeling costs would rise even more.

In *unsupervised learning*, we assume that we are given data $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n$ without corresponding outputs. The goal then is to find interesting patterns or knowledge in the data. This goal is not as clearly defined as in supervised learning [136]. A common unsupervised learning task is for instance clustering. In modern deep learning, one is often concerned with finding a useful representation $h(\mathbf{x})$ of the input such that future downstream tasks (e.g., classification) become simpler.

In *reinforcement learning* the goal is to maximize reward or minimize punishment. In the standard setting, it is assumed that an agent acts in an environment and the reward or punishment is provided by the environment [140]. While reinforcement learning is primarily associated with its successful application in gaming, as demonstrated by [141, 142], it has also been employed for imitating human behaviors and enforcing safety mechanisms in large language models [143, 144].

Note that not all learning tasks fit neatly into these categorizations. For instance, as we have seen, in Domain Generalization, aside from the class label y_i, we also have access to the environment label env_i (see Section 2.2).

4.1.2. Learning Algorithms

There exists a plethora of different learning algorithms and keeping oversight seems a hopeless endeavor. However, any learning algorithm can be categorized along three dimensions: *Representation*, *evaluation*, and *optimization* [1]. Following [1], we examine these dimensions in more detail. Here, we restrict ourselves to the classification task.

Representation We need to define a formal language to represent classifiers, as computers require this formal representation to utilize them. We term the set of expressible classifiers as the *hypothesis space* or *function space*. A natural trade-off arises in relation to the power of the representation (hypothesis space): Less powerful representations, such as linear functions, lack the ability to depict complex functions like non-linear ones. However, in this case, the hypothesis space is comparably small making search tasks manageable. More complex representations (e.g. neural networks as in Section 4.4) have the capacity to depict very complex functions but also extend the hypothesis space significantly.

Evaluation Since our objective is to find a classifier in the hypothesis space that provides the best fit to the data (for more details see Section 4.5), we need some form of evaluation. An evaluation function (or scoring function) assigns higher values to classifiers when they exhibit a better fit for the data. Note that practitioners might choose a scoring function based on an intent they like to achieve. Another motivation to select a specific evaluation function could be its improvement of the optimization procedure.

Optimization To find the best classifier (according to the evaluation function) within the hypothesis space, we need a strategy to smartly navigate through the hypothesis space. In most relevant applications the hypothesis space is immensely huge, and brute-force search methods are therefore doomed to fail. In the realm of deep learning, the prevalent choice for optimization is a variant of gradient-based optimization that we detail in Subsection 4.4.3. Typically, there is no guarantee to find the best classifier. However, we might guarantee to find the best classifier compared to neighboring ones in the hypothesis space (for details see Subsection 4.4.3).

4.2. Maximum Likelihood Estimation

One of the most common methods to estimate the model parameters in deep learning is the *maximum likelihood method* which we will briefly discuss in the following (for more details see e.g., [145, Chapter 9.3] or [38, Chapter 5.5]). Assume that the data samples $\mathbf{x}_1, \ldots, \mathbf{x}_n$ we observe are independent draws from the data generating distribution P. Furthermore, assume that the unknown distribution P has a density p can be described via parameters $\boldsymbol{\theta}_\star$, i.e. $P = P_{\boldsymbol{\theta}_\star}$ and $p = p_{\boldsymbol{\theta}_\star}$. The *maximum likelihood method* allows us to estimate $\boldsymbol{\theta}_\star$ and is defined as

$$\begin{aligned} \boldsymbol{\theta}_{\mathrm{ML}} &= \arg\max_{\boldsymbol{\theta}} p_{\boldsymbol{\theta}}(\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}) && (4.2) \\ &= \arg\max_{\boldsymbol{\theta}} \prod_{i=1}^{m} p_{\boldsymbol{\theta}}(\mathbf{x}_i) && (4.3) \\ & && (4.4) \end{aligned}$$

The term $p_{\boldsymbol{\theta}}(\{\mathbf{x}_1, \ldots, \mathbf{x}_n\})$ is also called *likelihood* or *likelihood-function*. The maximum likelihood estimator has some nice statistical properties that we do not detail here (see e.g., [145, Chapter 9.4] for more details). Since the arg max does not change when the logarithm is applied (due to monotonicity) or when it is multiplied with a positive constant, we obtain

$$\begin{aligned} \boldsymbol{\theta}_{\mathrm{ML}} &= \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i) && (4.5) \\ &= \arg\max_{\boldsymbol{\theta}} \frac{1}{m} \sum_{i=1}^{m} \log p_{\boldsymbol{\theta}}(\mathbf{x}_i) && (4.6) \end{aligned}$$

It is common to optimize the log-likelihood instead of the likelihood since this alleviates numerical instabilities (e.g. numerical underflow) and is often considered to be more convenient [38, Chapter 5.5].

In the infinite data regime, we get

$$\boldsymbol{\theta}_{\mathrm{ML}} = \arg\max_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{X} \sim P_{\boldsymbol{\theta}_\star}} \log p_{\boldsymbol{\theta}}(\mathbf{X}) \qquad (4.7)$$

The Kullback-Leibler (KL) divergence between $p_{\boldsymbol{\theta}_\star}$ and $p_{\boldsymbol{\theta}}$ is

$$\begin{aligned} D_{\mathrm{KL}}(p_{\boldsymbol{\theta}_\star} \| p_{\boldsymbol{\theta}}) &= \mathbb{E}_{\mathbf{X} \sim P_{\boldsymbol{\theta}_\star}} [\log p_{\boldsymbol{\theta}_\star}(\mathbf{X}) - \log p_{\boldsymbol{\theta}}(\mathbf{X})] && (4.8) \\ &= \mathrm{const} - \mathbb{E}_{\mathbf{X} \sim P_{\boldsymbol{\theta}_\star}} [\log p_{\boldsymbol{\theta}}(\mathbf{X})] && (4.9) \end{aligned}$$

Therefore, we can interpret the maximum likelihood estimation in Equation 4.7 as minimizing the KL divergence between the true density $p_{\boldsymbol{\theta}_\star}$ and the learned density $p_{\boldsymbol{\theta}}$. The KL divergence is 0 if and only if both distributions are equal. Therefore, we can conclude that finding the global maximum in Equation 4.7 leads to $p_{\boldsymbol{\theta}_\star}$.

The maximum likelihood estimator can be extended to conditional distributions as

$$\boldsymbol{\theta}_{\text{ML}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{m} \log p_{\boldsymbol{\theta}}(y_i \mid \mathbf{x}_i) \tag{4.10}$$

This is the basis for most supervised learning algorithms. We discuss this in more depth in Subsection 4.4.2.

4.3. Information Theoretical Concepts

In this work, we extensively use information theoretical concepts. This section provides a brief introduction to the most crucial concepts. Here, we assume that all random variables are continuous. The definitions and properties of this section are sourced from [146] and [38, Chapter 3.13]. We first define the differential entropy which should not be confused with the discrete entropy.

Definition 12 (Differential Entropy)**.** *The **differential entropy** of a continuous random variable X is defined as*

$$H(X) = \mathbb{E}_X[-\log p(X)] = -\int p(x)\log p(x)dx \tag{4.11}$$

Equivalently, we denote the entropy as $H(p)$. This definition and all the following ones extend to multiple random variables. In contrast to the discrete entropy, the differential entropy is not bounded from below. Similarly, we can define the *conditional differential entropy*.

Definition 13 (Conditional Differential Entropy)**.** *The **conditional differential entropy** is defined as*

$$H(X \mid Y) = -\mathbb{E}_{X,Y}[-\log p(X \mid Y)] = -\int p(x,y)\log p(x \mid y)dxdy. \tag{4.12}$$

One can show the property $H(X) \leq H(X \mid Y)$. This implies that the entropy might decrease when additional information (here represented by Y) is available. The *Kullback-Leibler (or KL) divergence* is a measure used to quantify the difference between two probability distributions by comparing their densities.

Definition 14 (KL-Divergence)**.** *The **Kullback-Leibler (or KL) divergence** is defined as*

$$D_{KL}(p\|q) = \mathbb{E}_{X\sim P}\left[\log\frac{p(X)}{q(X)}\right] = \int p(x)\log\frac{p(x)}{q(x)}dx \tag{4.13}$$

The KL divergence is always greater than or equal to 0 and is exactly equal to 0 if and only if $p = q$ almost everywhere. The KL divergence is asymmetric implying that the order in which it is expressed matters. Another important quantity is the *cross-entropy* from which important loss functions can be derived (see Subsection 3.8.2).

Definition 15 (Cross-Entropy)**.** *The **cross-entropy** between two distributions p, q is defined as*

$$H(p,q) = -\mathbb{E}_{X\sim P}[\log q(X)] = -\int p(x)\log q(x)dx \tag{4.14}$$

The quantity in Definition 15 is just the sum of the entropy of p and the KL divergence between p and q, namely

$$H(p,q) = H(p) + D_{\mathrm{KL}}(p\|q) \tag{4.15}$$

Therefore, we obtain

$$\min_q H(p,q) = \min_q D_{\mathrm{KL}}(p\|q) \tag{4.16}$$

This implies that if we find a q that minimizes $H(p,q)$ for a fixed p, then we achieve $p = q$ almost everywhere. An important measure for us is the *mutual information* that quantifies the amount of information two random variables carry about each other.

Definition 16 (Mutual Information)**.** *The **mutual information** $I(X,Z)$ between two random variables with joint density $p_{X,Y}$ is defined as*

$$I(X;Y) = D_{KL}(p_{X,Y}\|p_Xp_Y) \tag{4.17}$$

The mutual information between X and Y is equal to 0 if and only if X and Y are independent, which means that they do not contain any information about each other. This property can be understood through the definition via the KL-divergence: when the KL divergence is equal to 0, we obtain $p_{X,Y} = p_Yp_Y$. From the definition of mutual information, we can easily obtain

$$I(X;Y) = H(X) - H(X\mid Y) = H(Y) - H(Y\mid X) \tag{4.18}$$

Likewise, we can define the *conditional mutual information.*

Definition 17 (Conditional Mutual Information)**.** *The **conditional mutual information** is defined as*

$$I(X;Y\mid Z) = \int p_Z(z)D_{KL}(p_{X,Y|Z=z}\|q_{X,Y|Z=z})dz \tag{4.19}$$

Also here, we obtain that $I(X;Y\mid Z) = 0$ is equivalent to $X \perp Y \mid Z$.

4.4. Neural Networks

Feedforward Neural networks, also known as *Multi-layer perceptron* or *artificial neural networks*, are mathematical models that are loosely inspired by processes in the mammalian brain. In the following section, we introduce them along the categorization representation (Neural networks), evaluation (Loss functions), and optimization (Gradient-descent) introduced in Subsection 4.1.2. We base our terminology and definitions on [38, Chapter 6].

4.4.1. Neural Networks

A *feed-forward neural network* f is a mapping that consists of a sequence of operations

$$f(\mathbf{x}) = (f_n \circ f_{n-1} \cdots \circ f_1)(\mathbf{x}) \tag{4.20}$$

where the single operations f_i are termed layers. f_1 is called the *input layer*, f_n the *output layer* and all other layers are called *hidden layers*. The length n of this sequence is called the *depth* of the model. Each layer f_i consists of the same two building blocks: An affine transformation $A\mathbf{x} + b$ followed by a non-linear mapping $\mathbf{a}$ that acts independently on its input dimensions:

$$f_i(\mathbf{x}') = \mathbf{a}(A\mathbf{x}' + b) \tag{4.21}$$

Here, $A \in \mathbb{R}^{n \times m}$ is a real-numbered matrix, and $b \in \mathbb{R}^m$ a real vector. The entries in A are called *network weights* (or just *weights*) and the vector b is called *bias*. The matrix $A \in \mathbb{R}^{n \times m}$ of layer i is said to have *width* m. The non-linear mapping $\mathbf{a}$ is required to act independently on the input dimensions, i.e.

$$\mathbf{a}(\mathbf{x}) = [a_1(x^1), a_2(x^2), \ldots, a_n(x^n)] \tag{4.22}$$

Here a_i are non-linear functions that are called *activation functions*. Different activation functions have been proposed with different properties (see for instance Figure 4.1). One of the most common ones is the *ReLU (rectified linear unit)* as in Figure 4.1b. The parameters that describe the matrices A, biases b, and activation function $\mathbf{a}$ in all layers are denoted as $\boldsymbol{\theta}$. The network parameterized by $\boldsymbol{\theta}$ is referred to as $f_{\boldsymbol{\theta}}$.

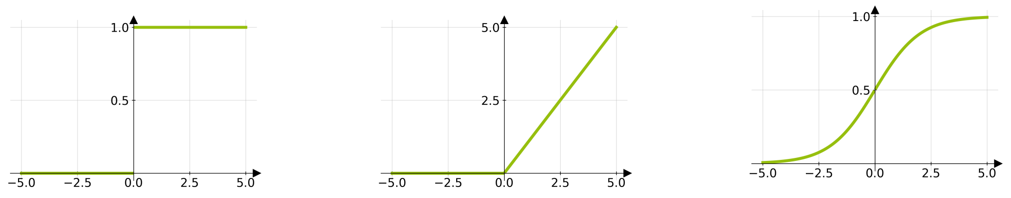
The choice of the output layer depends on the modality of the target variable (e.g., continuous vs. discrete). In a regression task, the output layer typically takes the form of a linear mapping. For binary classification, the sigmoid function as shown in Figure 4.1c can be employed to predict $P(Y \mid \mathbf{X})$ where Y is binary. In classification problems where the target value can take several values, the *softmax function* is commonly chosen and it takes the following form:

$$\text{softmax}(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)} \tag{4.23}$$

for inputs z. We then estimate $P(Y = c \mid \mathbf{X} = \mathbf{x})$ for class c via $\text{softmax}(z)_c$ where z is the output of all previous layers applied to $\mathbf{x}$. Note that the denominator ensures that all class probabilities add up to 1. For more details see [38, Chapter 6.2]

This type of network is called a feed-forward network because information flows from the input layer one-way through the hidden layers to the output layer [38, Chapter 6]. Feed-forward neural networks can be extended to include feedback connections in which case they are called recurrent neural networks (RNNs) [38, Chapter 6]. Specific restrictions to the weights of the network (e.g., in Convolutional Neural Networks) or combining different neural networks into a new function (e.g. as in Neural Turing Machines [147]) are referred to as network architectures. Specific architectures, such as the transformer model [148], have often been responsible for significant advancements in deep learning.

Neural networks serve as function approximators, making it crucial to define the space of learnable functions. First and foremost, excluding the non-linear activation function



(a) Heaviside step function

$$\mathbb{1}[x \geq 0]$$

(b) Rectified linear unit (ReLU)

$$y = \max(0, x)$$

(c) Sigmoid

$$y = \frac{1}{1 + e^{-x}}$$

Figure 4.1. Three activation functions for neural networks.

from the neural network would result in a purely linear model. In such cases, the neural network can represent only linear mappings, limiting its capabilities significantly. How much does the expressive power of a neural network increase when we use non-linear activation functions? It turns out that with various non-linear activation functions (including those in Figure 4.1), a neural network can approximate any continuous function on a closed and bounded subset of $\mathbb{R}^n$ provided that the network has at least one hidden layer with sufficient width. This property was shown by [149] for a class of activation functions, including the sigmoid and Heaviside step function *and* later extended by [150] for ReLU. This result is also referred to *universal approximation theorem* for neural networks. While we know that neural networks are theoretically highly expressive, in practice, we do not know the required network size to express the function we are seeking.

The name *neural networks* and *neurons* is derived from their biological inspiration, specifically from the synaptic connections where neurons send signals to other neurons. For one layer $\mathbf{o} = \mathbf{a}(A\mathbf{x} + b)$ with output $\mathbf{o}$, we can interpret each output dimension in $\mathbf{o} = o^1, \ldots, o^m$ as a weighted input-sum plus some bias fed into an activation function:

$$\mathbf{a}(A\mathbf{x} + b))_i = a_i \left(\sum_{j=1}^{n} A_{ij}\mathbf{x}_j + b_i \right) \tag{4.24}$$

Considering the Heaviside step function (as shown in Figure 4.1a) provides a biological explanation. If the weighted inputs, i.e. $\sum_{j=1}^{n} A_{ij}\mathbf{x}^j$, exceed a certain threshold of $-b_i$, neuron i is activated and fires. This resembles a simplified biological neuron.

4.4.2. Loss functions

There exists an abundance of different loss functions that serve as evaluation functions in deep learning. Most of these evaluation functions are the negative log-likelihood of the model distribution [38, Chapter 6.2]. Minimizing this quantity is equivalent to minimizing the cross-entropy between training and predicted distribution or maximizing the likelihood (see Section 4.2 and Section 4.3).

Conditional Log-Likelihood In supervised learning, the cross-entropy has the following generic form

$$H(p_{Y|\mathbf{x}}, p_{\boldsymbol{\theta}}) = -\mathbb{E}_{(\mathbf{X},Y)\sim p}[\log p_{\boldsymbol{\theta}}(Y \mid \mathbf{X})] \tag{4.25}$$

where p is the distribution from where the data origins [38, Chapter 6.2]. Depending on the assumption of $p_{\boldsymbol{\theta}}(y \mid x)$, the cross-entropy yields a different loss or cost function.

For a regression task, it is often assumed that the model distributions follow a normal distribution, i.e. $p_{\boldsymbol{\theta}}(y \mid \mathbf{x}) = \mathcal{N}(y; f_{\boldsymbol{\theta}}(\mathbf{x}), \mathbf{I})$. In this case, we recover the L2-Loss

$$-\mathbb{E}_{(\mathbf{X},Y)\sim p}[\log p_{\boldsymbol{\theta}}(Y \mid \mathbf{X})] = \frac{1}{2}\mathbb{E}_{\mathbf{X},Y}[\|Y - f_{\boldsymbol{\theta}}(\mathbf{X})\|_2^2] - \text{const} \tag{4.26}$$

up to a scaling factor and some constant [38, Chapter 6.2]. Different assumptions on $p_{\boldsymbol{\theta}}(y \mid \mathbf{x})$ result in different loss functions. For instance, if one assumes $p_{\boldsymbol{\theta}}(y \mid \mathbf{x})$ is a Laplacian distribution for all $\mathbf{x}$, one can recover the L1-Loss.

For the classification task, it is commonly assumed that the model distribution $p_{\boldsymbol{\theta}}(y \mid \mathbf{x})$ follows a *multinoulli* or *categorical* distribution. In this case, the cross-entropy is

$$H(p_{Y|\mathbf{x}}, p_{\boldsymbol{\theta}}) = -\mathbb{E}_{(\mathbf{X},Y)\sim p}[\log p_{\boldsymbol{\theta}}(Y \mid \mathbf{X})] = -\mathbb{E}_{(\mathbf{X},Y)\sim p}\left[\log \prod_{c=1}^{C} p_{\boldsymbol{\theta}}(c \mid \mathbf{X})^{\mathbb{1}[Y=c]}\right] \tag{4.27}$$

$$= \mathbb{E}_{(\mathbf{X},Y)\sim p}\left[\sum_{c=1}^{C} \mathbb{1}[Y = c] \log p_{\boldsymbol{\theta}}(c \mid \mathbf{X})\right] \tag{4.28}$$

where $Y \in \{1, 2, \ldots, C\}$ and C is the amount of classes to predict. On a finite dataset $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, this entity is

$$\sum_{i=1}^{n}\left(\sum_{c=1}^{C} \mathbb{1}[y_i = c] \cdot \log p_{\boldsymbol{\theta}}(c \mid \mathbf{x}_i)\right) \tag{4.29}$$

To model $p_{\boldsymbol{\theta}}$ we could employ the softmax function in the output layer as explained above. It is important to note that the cross-entropy loss is a strictly *proper scoring rule* which means that if we find the global optimum, then the learned model distribution faithfully models the true distribution [151]. This implies, in particular, that we can trust the uncertainties of the model's predictions.

4.4.3. Gradient-Descent and Backpropagation

Gradient-Descent The standard approach to minimize the loss of neural networks is gradient-descent or a variant thereof [38, 152]. The concept behind gradient-descent involves evaluating a function at a specific point and calculating its gradient. The gradient provides directional guidance for either decreasing or increasing the function's output. Moving a small step in the negative direction of the gradient allows us to decrease the function's output. See Figure 4.2 for a visual explanation. Gradient-descent does not guarantee

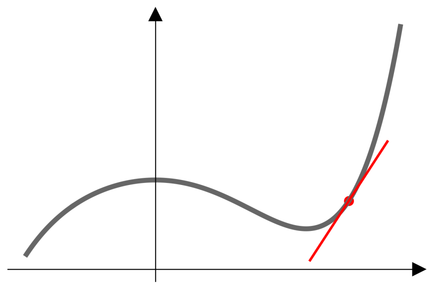


Figure 4.2. Illustration of gradient-descent. The black line represents the function under consideration, and the red line shows the linear extrapolation of the functions' gradient at this specific point. Moving a small step in the negative direction of the gradient allows us to locate a point where the function is smaller than at the red point.

achieving the global minimum, but it ensures finding a local minimum under suitable conditions. Global minimums are only guaranteed for convex functions.

In the context of deep learning, we would like to minimize the training error

$$\mathcal{L}(\boldsymbol{\theta}) = \mathcal{R}_{\text{train}}[f_{\boldsymbol{\theta}}] = \sum_{\mathbf{x},y \in \mathcal{D}_n} c(f_{\boldsymbol{\theta}}(x), y) \tag{4.30}$$

with respect to $\boldsymbol{\theta}$ where $\mathcal{D}_n$ is the training dataset of size n. Since this error is a sum, the derivative also decomposes into a sum

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \sum_{\mathbf{x},y \in \mathcal{D}_n} \nabla_{\boldsymbol{\theta}} c(f_{\boldsymbol{\theta}}(x), y) \tag{4.31}$$

This eases the computation of the overall gradient. We explain in Section 4.5 the relation between the training error $\mathcal{R}_{\text{train}}[f_{\boldsymbol{\theta}}]$ and the error on the whole distribution. In the following, we give a heuristical derivation of the gradient-descent approach. With the Taylor expansion we obtain

$$\mathcal{L}(\boldsymbol{\theta}) \approx \mathcal{L}(\boldsymbol{\theta}_0) + (\boldsymbol{\theta} - \boldsymbol{\theta}_0)^T \nabla \mathcal{L}(\boldsymbol{\theta}_0) \tag{4.32}$$

for $\boldsymbol{\theta}$ close to $\boldsymbol{\theta}_0$ and therefore

$$\mathcal{L}(\boldsymbol{\theta}_0 - \alpha \nabla \mathcal{L}(\boldsymbol{\theta}_0)) \approx \mathcal{L}(\boldsymbol{\theta}_0) - \alpha (\nabla \mathcal{L}(\boldsymbol{\theta}_0))^T \nabla \mathcal{L}(\boldsymbol{\theta}_0) \tag{4.33}$$

for small $\alpha > 0$. If $\alpha > 0$ is small enough, we obtain

$$\mathcal{L}(\boldsymbol{\theta}_0) - \mathcal{L}(\boldsymbol{\theta}_0 - \alpha \nabla \mathcal{L}(\boldsymbol{\theta}_0)) \approx +\alpha (\nabla \mathcal{L}(\boldsymbol{\theta}_0))^T \nabla \mathcal{L}(\boldsymbol{\theta}_0) \geq 0 \tag{4.34}$$

Therefore, we could heuristically argue that

$$\mathcal{L}(\boldsymbol{\theta}_0) > \mathcal{L}(\boldsymbol{\theta}_0 - \alpha \nabla \mathcal{L}(\boldsymbol{\theta}_0)) \tag{4.35}$$

if $\alpha > 0$ is small enough and $\nabla \mathcal{L}(\boldsymbol{\theta}_0) \neq 0$. So, if we update the parameters $\boldsymbol{\theta}$ in the direction of the negative gradient, we minimize the risk provided that $\mathcal{L}(\boldsymbol{\theta}_0)$ is not already a minimum or a saddle point *and* the update step α is small enough.

In deep learning, we usually do not compute the gradient of the whole dataset at once, but on a sub-sample (also called *mini-batch*). Choosing a mini-batch over the whole dataset offers several advantages, including the fact that computing the gradient on the whole dataset is often unfeasible given the enormous size of modern datasets (for a more depth-analysis on minibath gradient-descent see [38, Chapter 8]). *Stochastic Gradient-descent (SGD)* is a crucial gradient-descent variant in deep learning that employs mini-batches. Before we can update the gradients on a mini-batch in SGD, we need to specify a *learning rate schedule* $\alpha_1, \alpha_2, \ldots$ which determines the step size for gradient updates at each iteration. Typically, the step size decreases, i.e.$\alpha_1 \geq \alpha_2 \geq \ldots$. Choosing a learning rate schedule is a delicate task. If α_j becomes too small for $j > i$, convergence to a minimum may be unachievable within reasonable time frames. In contrast, if α_j is too big for all $j > i$, we risk missing the minima at each iteration. In addition to the learning parameter, we need an *initial parameter configuration* $\boldsymbol{\theta}$ from where we start. Selecting an appropriate initial parameter configuration is a research topic on its own. SGD then proceeds as follows, starting at $k = 1$ and continuing until a stop criterion is met:

(1) Sample mini-batch $\{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(m)}, y^{(m)})\}$ of m examples from the training set $\mathcal{D}_n$

(2) Compute gradient estimate

$$\text{grad} = \frac{1}{m} \nabla_{\boldsymbol{\theta}} \sum_{i=1}^{m} c(f_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}), y^{(i)}) \tag{4.36}$$

(3) Update parameters: $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \alpha_k \text{grad}$; And update current step/iteration: $k \leftarrow k + 1$

SGD is often enhanced by incorporating the method of momentum where gradients are accumulated to improve optimization. Consider [38, Chapter 8] for more details on SGD or momentum approaches.

Backpropagation We showed how gradient-descent can lead to an improvement in terms of the error $\mathcal{L}(\boldsymbol{\theta})$. However, we did not explain how $\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$ can be computed efficiently. Two apparent solutions might come to mind. Firstly, we could approximate the gradient of a function $\mathcal{L}(\theta)$ by computing the finite difference of its partial derivatives with respect to each parameter θ_i:

$$\frac{\partial \mathcal{L}(\theta)}{\partial \theta_i} \approx \frac{\mathcal{L}(\theta + h e_i) - \mathcal{L}(\theta)}{h} \tag{4.37}$$

where $h > 0$ is very small positive value, and e_i is a vector with the same shape as θ that contains a 1 at entry i and 0 elsewhere. This procedure is also known as *numerical differentiation*. However, it can be imprecise due to approximation errors and becomes intractable for large neural networks. Secondly, we could employ the chain rule and compute an exact closed-form solution of the partial derivatives, commonly referred to as *symbolic differentiation*. However, for complex functions, a closed-form solution of the partial derivatives can become prohibitively memory-intensive, as illustrated in the example below.

An effective solution that addresses these issues is *automatic differentiation* or *autodiff*, which comes in two modes: Forward and reverse [38, Chapter 6.5]. The widely used *backpropagation* (also called backprop) algorithm is an instance of the reverse mode. The

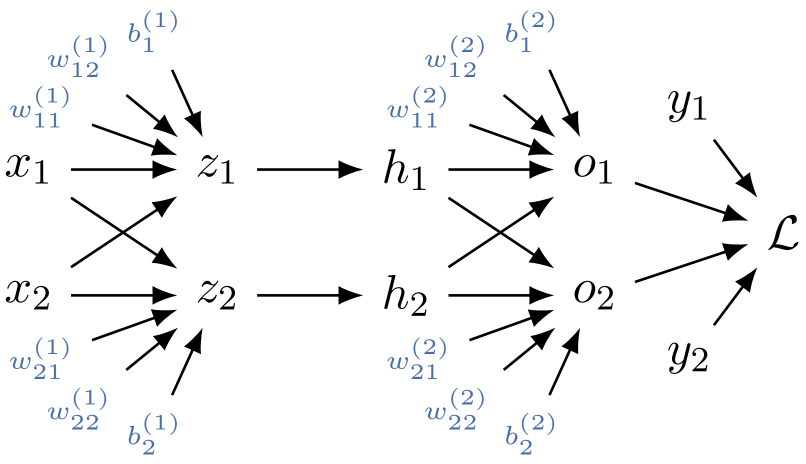


Figure 4.3. Computational graph of example. The graph is due to [153].

backpropagation algorithm computes the gradients by recursively applying the multivariate chain rule. In a neural network, a computation is propagated through the network in a forward manner (also referred to as *forward pass*), starting from the first layer to the last. The backpropagation algorithm operates backward starting from the last layer, and going back to the first one (also known as *backward pass*). We will briefly discuss the backpropagation algorithm with an example and refer for details to [38, Chapter 6.5].

Here we use the same computational graph as in [153] (see Figure 4.3). First, let us briefly revisit the multi-variate chain rule. Consider a function $f\colon \mathbb{R}^m \to \mathbb{R}$, and functions $g_i\colon \mathbb{R} \to \mathbb{R}$ for $i = 1, \ldots, m$. Then we we can compute the derivative of the composition $f(g_1(x), \ldots, g_k(x))$ as

$$\frac{\partial f}{\partial x} = \sum_{j=1}^{m} \frac{\partial f}{\partial g_j} \frac{\partial g_j}{\partial x} \tag{4.38}$$

Figure 4.3 represents a computational graph where nodes without parents are fixed scalars and nodes with parents represent functions (or computations) with incoming nodes as the function's arguments. The forward pass in the computation graph in Figure 4.3 is defined via

$$z_i = \sum_{j=1}^{2} w_{ij}^{(1)} x_j + b_i^{(1)}, \qquad i = 1, 2 \tag{4.39}$$

$$h_i = a(z_i), \qquad i = 1, 2 \tag{4.40}$$

$$o_k = \sum_{i=1}^{2} w_{ki}^{(2)} h_i + b_k^{(2)}, \qquad k = 1, 2 \tag{4.41}$$

$$\mathcal{L} = \frac{1}{2} \sum_{k=1}^{2} (y_k - o_k)^2 \tag{4.42}$$

where a is an activation function. The backward pass then takes the following form:

	Chain Rule	Backprop	Symbolic Derivation
$\frac{\partial \mathcal{L}}{\partial \mathcal{L}}$	$= 1$	$=1$	$=1$
$\frac{\partial \mathcal{L}}{\partial y_k}$	$= \frac{\partial \mathcal{L}}{\partial \mathcal{L}} \frac{\partial \mathcal{L}}{\partial y_k}$	$= \frac{\partial \mathcal{L}}{\partial \mathcal{L}} (o_k - y_k)$	$=1 \cdot (o_k - y_k)$
$\frac{\partial \mathcal{L}}{\partial w_{ki}^{(2)}}$	$= \frac{\partial \mathcal{L}}{\partial y_k} \frac{\partial y_k}{\partial w_{ki}^{(2)}}$	$= \frac{\partial \mathcal{L}}{\partial y_k} h_i$	$=(o_k - y_k) h_i$
$\frac{\partial \mathcal{L}}{\partial b_k^{(2)}}$	$= \frac{\partial \mathcal{L}}{\partial y_k} \frac{\partial y_k}{\partial b_k^{(2)}}$	$= \frac{\partial \mathcal{L}}{\partial y_k} \cdot 1$	$=(o_k - y_k) \cdot 1$
$\frac{\partial \mathcal{L}}{\partial h_i}$	$= \sum_k \frac{\partial \mathcal{L}}{\partial y_k} \frac{\partial y_k}{\partial h_i}$	$= \sum_k \frac{\partial \mathcal{L}}{\partial y_k} w_{ki}^{(2)}$	$= \sum_k (o_k - y_k) w_{ki}^{(2)}$
$\frac{\partial \mathcal{L}}{\partial z_i}$	$= \frac{\partial \mathcal{L}}{\partial h_i} \frac{\partial h_i}{\partial z_i}$	$= \frac{\partial \mathcal{L}}{\partial h_i} a'(z_i)$	$= \left(\sum_k (o_k - y_k) w_{ki}^{(2)} \right) a'(z_i)$
$\frac{\partial \mathcal{L}}{\partial w_{ij}^{(1)}}$	$= \frac{\partial \mathcal{L}}{\partial z_i} \frac{\partial z_i}{\partial w_{ij}^{(1)}}$	$= \frac{\partial \mathcal{L}}{\partial z_i} x_j$	$= \left(\sum_k (o_k - y_k) w_{ki}^{(2)} \right) a'(z_i) x_j$
$\frac{\partial \mathcal{L}}{\partial b_i^{(1)}}$	$= \frac{\partial \mathcal{L}}{\partial z_i} \frac{z_i}{b_i^{(1)}}$	$= \frac{\partial \mathcal{L}}{\partial z_i} \cdot 1$	$= \left(\sum_k (o_k - y_k) w_{ki}^{(2)} \right) a'(z_i)$

Recursive Computation

One can easily verify that by recursively applying the chain rule and using the computed gradient from the previous layers to instantly compute the gradients, we drastically reduce the symbols to keep track of. Therefore, the backpropagation algorithm is a much more elegant and memory-efficient variant compared to symbolic differentiation.

Consider a topological ordering of the computation graph $v_1, \dots, v_N$ representing the neural network. In this ordering, parents precede their children. Generally, the backpropagation algorithm starts with the forward pass, where values at each node in the computational graph are stored. Subsequently, a simplified version of the backpropagation algorithm proceeds in the following way [38, 153]

(1) Set $\frac{\partial \mathcal{L}}{\partial \mathcal{L}} = 1$

(2) For $i = N-1, \dots, 1$:

(2.1) For the computational node v_i compute the gradient via the multi-variate chain rule

$$\frac{\partial \mathcal{L}}{\partial v_i} = \sum_{l \in ch(v_i)} \frac{\partial \mathcal{L}}{\partial v_l} \frac{\partial v_l}{\partial v_i} \tag{4.43}$$

For more details on the backpropagation algorithm (and in particular on a vectorized version) see [38, Chapter 6.5].

4.5. Generalization

The main goal of machine learning is to generalize knowledge from observed training data [1, 2]. This delineates machine learning from pure optimization [38]. In the following, we formalize the generalization task in the supervised learning (and in-distribution) setting and discuss why it is so difficult. Afterwards, we analyze the generalization error in more detail from a theoretical as well as a practical point of view. We conclude this section by discussing the differences between generalization in the in-distribution (ID) and out-of-distribution (OOD) setting.

4.5.1. The Generalization Task

The goal of supervised learning is to find a function f in a function family $\mathcal{F}$ that minimizes the *expected error* or *risk*

$$\mathcal{R}[f] = \mathbb{E}_{Y,\mathbf{X}}[c(Y, f(\mathbf{X})] \tag{4.44}$$

where c is some loss function that depends on the task at hand. For instance, $c(y, f(\mathbf{x})) = \|y - f(\mathbf{x})\|_2^2$ might be the squared Euclidean distance. However, given a predictive model f, its risk is not available in practice, since we lack direct access to the true distribution $P(\mathbf{X}, Y)$. What we have instead are independent samples

$$\mathcal{D}_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\} \tag{4.45}$$

or simply $\mathcal{D}$, drawn from the true distribution. The training data allows us to determine the *empirical risk* or *training error* for a given model f

$$\mathcal{R}_{\text{train}}[f] = \frac{1}{n}\sum_{i=1}^{n} c(y_i, f(\mathbf{x}_i)) \tag{4.46}$$

We denote the function in $\mathcal{F}$ that minimizes the risk as

$$f^\star = \arg\min_{f\in\mathcal{F}} \mathcal{R}[f] \tag{4.47}$$

and likewise the function in $\mathcal{F}$ that minimizes the empirical risk on $\mathcal{D}$ as

$$f^\star_{\mathcal{D}} = \arg\min_{f\in\mathcal{F}} \mathcal{R}_{\text{train}}[f] \tag{4.48}$$

The *empirical risk minimization (ERM) principle* [154] requests to learn or find a function that minimizes the empirical risk in Equation 4.46 in order to minimize the risk Equation 4.44. Put a little differently, it demands $f^\star_{\mathcal{D}}$ to closely match $f^\star$. Typically, the *learning procedure* or *training algorithm* finds only a function, denoted by $\hat{f}_{\lambda,\mathcal{D}} \in \mathcal{F}$, that achieves sub-optimal training error. The symbol λ describes the specifics of the learning procedure, such as network architecture, learning rate, or regularization/penalization term. In such cases, this process often involves seeking an optimum within a subspace $\mathcal{F}_\lambda \subset \mathcal{F}$ rather than within the entire function space $\mathcal{F}$. For instance, in Ridge regression the regularization term λ confines $\mathcal{F}$ to a subspace $\mathcal{F}_\lambda$ [155, Chapter 2.3].

The function f_{Bayes} that minimizes the risk is called *Bayes rule*. In classification, the *Bayes rule* or *Bayes classifier* is given by

$$f_{\text{Bayes}}(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} P(Y = y \,|\, \mathbf{X} = \mathbf{x}) \tag{4.49}$$

We can therefore bound the risk of any function from below and formulate the supervised learning task to find a function $f \in \mathcal{F}$ that minimizes the *excess Bayes risk* $\mathcal{R}[f] - \mathcal{R}[f_{\text{Bayes}}] \geq 0$.

A desirable property of a learning algorithm is that as the training set size n increases, the expected risk of $\hat{f}_{\lambda,\mathcal{D}}$ should closely match the Bayes risk $\mathcal{R}[f_{\text{Bayes}}]$ with high probability. This is called Bayes consistency [156]. One could think that if we make $\mathcal{F}$ as powerful as possible, e.g., encompass all possible functions, then Bayes consistency is guaranteed. However, this is not the case. This is easily seen if a learning algorithm just remembers the training data and predicts arbitrarily on new input. In this case, the expected Risk will be high, and the training error zero (see also [156] for an elaboration on this example). It turns out that if we restrict the space of admissible functions $\mathcal{F}$, Bayes-consistency can be achieved (see e.g., [156] for details).

Restricting the space of admissible functions is not enough. An implication of the *no-free-lunch theorem* is that for any classifier f that achieves zero empirical risk, there is a substantial amount of distributions from which the training set could have been drawn and where f has no better risk than random guessing [156, 157]. Is generalization therefore a hopeless endeavour? If we do not make assumptions on the distribution, the answer to this question is *yes*. But if we impose assumptions on the distributions from which the training set has been drawn (e.g., the distribution is smooth), then we can achieve theoretical generalization results (see e.g., [156, 158]).

In total, we require assumptions on the function space and the distribution P where we want to generalize. The restrictions that we put on the training algorithm $\hat{f}_{\lambda,\mathcal{D}}$ are sometimes called *inductive bias* and we discuss its role below in more detail.

4.5.2. Risk Decompositions

As stated above, we want the excess Bayes risk $\mathcal{R}[\hat{f}_{\lambda,\mathcal{D}}] - \mathcal{R}[f_{\text{Bayes}}] \geq 0$ to be as small as possible. It turns out that we can analyze this difference in more detail and learn about separate components that add up to the difference.

The Estimation-Approximation Trade-Off A simple transformation into *estimation error* and *approximation error* is achieved via (see also [156] or [155])

$$\mathcal{R}[f^\star_{\mathcal{D}}] - \mathcal{R}[f_{\text{Bayes}}] = \underbrace{(\mathcal{R}[f^\star_{\mathcal{D}}] - \mathcal{R}[f^\star])}_{\text{estimation error}} + \underbrace{(\mathcal{R}[f^\star] - \mathcal{R}[f_{\text{Bayes}}])}_{\text{approximation error}} \tag{4.50}$$

The estimation error is a random quantity influenced by the training set. It indicates the sensitivity of the learning algorithm to various draws of training sets. It furthermore signifies the additional error arising from optimizing the empirical risk rather than the actual risk. On the other hand, the approximation error is the systematic error due to the lack of expressiveness in $\mathcal{F}$. For instance, if the function class $\mathcal{F}$ is limited (e.g. only linear functions), then a non-linear Bayes rule cannot be fitted. For more details on this decomposition see also [156].

As we typically obtain $\hat{f}_{\lambda,\mathcal{D}}$ instead of $f^{\star}_{\mathcal{D}}$, [159] proposed an extension of the estimation-approximation decomposition by introducing an optimization error. This error term quantifies the discrepancy arising from the inability to find the minimum within $\mathcal{F}$. The decomposition is as follows:

$$\mathcal{R}[\hat{f}_{\lambda,\mathcal{D}}] - \mathcal{R}[f_{\text{Bayes}}] = \underbrace{\mathcal{R}[\hat{f}_{\lambda,\mathcal{D}}] - \mathcal{R}[f^{\star}_{\mathcal{D}}]}_{\text{optimization error}} + \underbrace{\mathcal{R}[f^{\star}_{\mathcal{D}}] - \mathcal{R}[f^{\star}]}_{\text{estimation error}} + \underbrace{(\mathcal{R}[f^{\star}] - \mathcal{R}[f_{\text{Bayes}}])}_{\text{approximation error}} \tag{4.51}$$

Bias-Variance Trade-Off It can be shown that the expected risk can also be decomposed in a *noise*, *bias* and *variance* term for the squared loss $c(y, f(x)) = (y - f(x))^2$

$$\underbrace{\mathbb{E}_{\mathcal{D}}[\mathcal{R}[\hat{f}_{\lambda,\mathcal{D}}]]}_{\text{expected risk}} = \underbrace{\mathbb{E}_{\mathbf{X},Y}[(Y - \overline{Y})^2]}_{\text{noise}} + \underbrace{\mathbb{E}_{\mathbf{X}}\left[\left(\mathbb{E}_{\mathcal{D}}[\hat{f}_{\lambda,\mathcal{D}}(\mathbf{X})] - \overline{Y}\right)^2\right]}_{\text{bias}}$$
$$+ \underbrace{\mathbb{E}_{\mathbf{X}}\left[\mathbb{E}_{\mathcal{D}}[(\hat{f}_{\lambda,\mathcal{D}}(\mathbf{X}) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\lambda,\mathcal{D}}(\mathbf{X})])^2]\right]}_{\text{variance}} \tag{4.52}$$

where $\overline{Y} = \mathbb{E}_{Y|\mathbf{X}}[Y]$ [160]. The noise term is the irreducible error or Bayes error that is only achieved for the Bayes rule. The bias term describes the systematic error resulting from limitations of the training procedure $\hat{f}_{\lambda,\mathcal{D}}$ to capture the complex relation between $\mathbf{X}$ and Y, independent of the training samples. This term indicates *underfitting*. In contrast, the variance term describes the sensitivity of the training procedure due to variations in the selection of the training set. When $\hat{f}_{\lambda,\mathcal{D}}$ is very sensitive to the training data selection, it indicates that $\hat{f}_{\lambda,\mathcal{D}}$ learns not the *signal* within the data, but rather the *noise*. When the training procedure captures the noise rather than the signal, it is termed *overfitting*. We will delve deeper into overfitting and underfitting below.

Interestingly, it has been demonstrated that a bias-variance trade-off exists for other loss functions c – in more general terms for any Bregman divergence – as well [161, 162]:

$$\mathbb{E}_{\mathcal{D}}[\mathcal{R}[\hat{f}_{\lambda,\mathcal{D}}]] = \underbrace{\mathbb{E}_{\mathbf{X},Y}[c(Y, \overline{Y})]}_{\text{noise}} + \underbrace{\mathbb{E}_{\mathbf{X}}\left[c(\overline{Y}, \overset{\circ}{f}_{\lambda}(\mathbf{X}))\right]}_{\text{bias}} + \underbrace{\mathbb{E}_{\mathbf{X}}\left[c(\overset{\circ}{f}_{\lambda}(\mathbf{X}), \hat{f}_{\mathcal{D},\lambda}(\mathbf{X}))\right]}_{\text{variance}} \tag{4.53}$$

Here $\overset{\circ}{f}_{\lambda}(\mathbf{X})$ represents a measure of centrality, also termed *centroid* and may vary based on the employed loss function[163, 164]. Notably, if we utilize the squared loss, we obtain $\overset{\circ}{f}_{\lambda} = \mathbb{E}_{\mathcal{D}}[\hat{f}_{\lambda,\mathcal{D}}]$ and the variance-bias trade-off in Equation 4.52 is recovered. For KL-divergence or Poisson regression, the centroid takes a different form [164]. For more details refer to [163, 164]

Bias-Variance and Estimation-Approximation Trade-Off The authors in [164] demonstrated that the expected estimation error can be decomposed as follows, when a bias-variance decomposition exists:

$$\underbrace{\mathbb{E}_{\mathcal{D}}[\mathcal{R}[f^\star_{\mathcal{D}}] - \mathcal{R}[f^\star]]}_{\text{exp. estimation error}} = \underbrace{\mathbb{E}_{\mathcal{D}}[\mathcal{R}[f^\star_{\mathcal{D}}] - \mathcal{R}[\mathring{f}_\lambda]]}_{\text{estimation variance}} + \underbrace{\mathcal{R}[\mathring{f}_\lambda] - \mathcal{R}[f^\star]}_{\text{estimation bias}} \tag{4.54}$$

Furthermore, they establish that the bias term could be broken down into the approximation error plus the estimation bias:

$$\underbrace{\mathbb{E}_{\mathbf{X}}[c(Y, \mathring{f}_\lambda)]}_{\text{bias}} = \underbrace{\mathcal{R}[f^\star] - \mathcal{R}[f_{\text{Bayes}}]}_{\text{approximation error}} + \underbrace{\mathcal{R}[\mathring{f}_\lambda] - \mathcal{R}[f^\star]}_{\text{estimation bias}} \tag{4.55}$$

Similarly, the variance term was shown to consist of the optimization error and estimation variance:

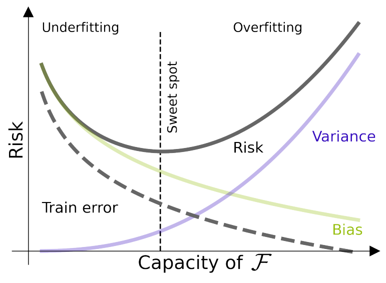
$$\underbrace{\mathbb{E}_{\mathbf{X}}\left[c(\mathring{f}_\lambda, \hat{f}_{\mathcal{D},\lambda}(\mathbf{X}))\right]}_{\text{variance}} = \underbrace{\mathbb{E}_{\mathcal{D}}\left[\mathcal{R}[\hat{f}_\lambda] - \mathcal{R}[f^\star_{\mathcal{D}}]\right]}_{\text{exp. optimization error}} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\mathcal{R}[f^\star_{\mathcal{D}}] - \mathcal{R}[\mathring{f}_\lambda]\right]}_{\text{estimation variance}} \tag{4.56}$$

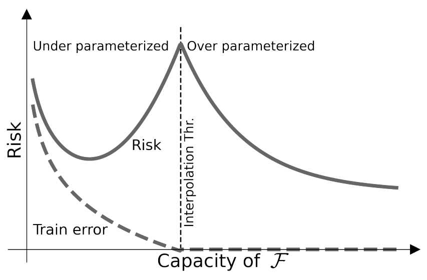
These decompositions provide insights into the components' contribution to the overall expected excess Bayes risk [164]:

$$\begin{aligned} \text{Exp. excess Bayes risk} &= \text{Exp. risk} + \text{Bayes error} && (4.57) \\ &= \text{Exp. opt. error} + \text{Exp. est. error} + \text{approx. error} && (4.58) \\ &= \underbrace{(\text{Exp. opt. error} + \text{est. variance})}_{\text{variance}} + \underbrace{(\text{est. bias} + \text{approx. error})}_{\text{bias}} && (4.59) \end{aligned}$$

Hence, the approximation-estimation trade-off and variance trade-off are not exactly the same but are closely related. Note that the expectation refers to random draws of the training set $\mathcal{D}$.

Underfitting and Overfitting The behavior of overfitting and underfitting is often associated with the complexity of the function space $\mathcal{F}$ as shown in Figure 4.4a. The more expressive $\mathcal{F}$, the lower the systematic error (bias), but the more sensitive is the result to the training data (high variance). This is the famous bias-variance trade-off and results in a U-shaped risk curve. A natural consequence is that the training error goes to zero if the function capacity is high enough. But at that point, we are already in the realm of overfitting: the risk is high while the training error is very small. The goal is then to find the sweet spot of being expressive, but not too expressive to get the best risk. More recently the phenomenon of *double descent* seems to challenge this conventional view. In particular, in deep neural networks it has been observed that with additional model complexity, the Risk curve experiences a second descent falling even below the minimum of the U-shaped curve [165]. This is illustrated in Figure 4.4b. While the classical view and phenomenon of double descent seem to contradict each other, a potential reconciling explanation lies in the inductive bias: Stochastic gradient-descent seems to prefer smoother functions over non-smoother functions and if we consider more complex models, we might find smoother functions compared to less complex models [166]. Smoothness seems to be an inductive bias that improves generalization for most real-world datasets.





(a) Conventional view on *bias-variance trade-off*. U-shaped risk curve that is composed of a bias and variance term and varies with the Capacity of $\mathcal{F}$.

(b) Illustration of *double descent* phenomenon. The risk curve experiences a second descent when the training algorithm can choose from a more powerful function class $\mathcal{F}$.

Figure 4.4. Bias-variance trade-off and double descent. Figure adapted from [166].

Curse of Dimensionality There are several reasons that might lead to overfitting. Above we discussed the role of the learning algorithms and inductive biases. But another reason stems from the peculiarities of the training set. Especially when the training set lacks sufficient information to enable the training algorithm to learn a predictive function. Therefore, we need to ask how many samples are enough to give a sufficiently big coverage of the true data distribution. Many inputs in machine learning are high dimensional, i.e. $\mathbf{x} \in \mathbb{R}^D$ with a large D. The *curse of dimensionality* refers to the observation that numerous problems exhibit distinct behaviors if more than just the geometrically interpretable dimensions of one, two, or three are considered. For instance, if we are required to evenly sample a unit interval such that the points have a distance of at most 10^{-2}. In this case, we need at least $(10^2)^1 = 10^2$ data points. Things change dramatically when we consider a higher dimensional unit hypercube. If we like to sample a 100-dimension hypercube evenly such that the distance between samples is at most 10^{-2}, we require at least $(10^2)^{100} = 10^{200}$ samples. Even if we had a trillion data samples 10^{12}, we would achieve only a tiny coverage of the total space (Example is due to [167]). This might indicate that learning in high dimensions is impossible. Fortunately, data samples are not uniformly distributed in space which makes learning even in high-dimensional problems possible – this is termed the *blessing of dimensionality* [168].

4.5.3. Model Selection and Estimating Generalization Error

Since we aim to find a function that generalizes well on new data, we actually try to achieve two related goals (see [158, Section 7.2]):

- find the best model, and
- estimate how it will generalize on new data

Training, Validation and Test Set The typical approach in machine learning to address these issues is to split all the available data into *training*, *validation*, and *test* sets. Different models (e.g., different network architectures) are then trained on the training set and evaluated on the validation set. In this procedure, the best model is chosen according to the empirical risk on the validation set. To estimate how the best model behaves under new data, it is evaluated on unseen test data. Here we assume training, validation, and test sets to be independent draws from the same distribution. A separate test set is necessary because the optimization process in the first step (finding the best model) could lead to overfitting on the validation set and therefore the empirical risk on the validation set might be overly optimistic about the risk. Note that if the test set is big enough, then we obtain convergence due to the law of large numbers (see also [155])

$$\mathcal{R}_{\text{test}}[\hat{f}_{\lambda,\mathcal{D}_n}] = \frac{1}{m_{\text{test}}} \sum_{(\mathbf{x},y)\in\mathcal{D}_{\text{test}}} c(y, \hat{f}_{\lambda,\mathcal{D}_n}(\mathbf{x})) \overset{m_{\text{test}}\to\infty}{\to} \mathcal{R}[\hat{f}_{\lambda,\mathcal{D}_n}] \tag{4.60}$$

where $\mathcal{D}_{\text{test}}$ is a set of m_{test} samples independent of training and validation samples. An important question is how to choose the proportion between training, validation, and test set. There is no ultimate answer to this question since we face a trade-off. For instance, if the training set is too small, then the training process result might not capture the signal in the data. On the other end, if the test set is too small, then our estimate of the risk might be imprecise. A common choice of practitioners is to select 60% of all data for training, 20% for validation, and 20% as test set. For more details on the overall procedure consult for instance [158] or [155].

Cross Validataion The training, validation, and test split approach is a very sound process to find the best model and estimate its risk. However, one problem is that the learning algorithm $\hat{f}_{\lambda,\mathcal{D}}$ only sees a proportion of all available data. This is particularly problematic if we only have little data at hand. In this case, the result might be very sensitive to how the data was split into training, validation, and test set. In a similar vein, we get an estimate for the risk of just one execution of the learning procedure $\mathcal{R}[\hat{f}_{\lambda,\mathcal{D}}]$ on one dataset D. However, we might like to fully evaluate the training procedure $\hat{f}_{\lambda,\cdot}$ on the whole data at hand.

M-fold cross-validation offers a solution to these problems. With this procedure, we split all available data into a training set $\mathcal{D}_n = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ and a test set that has no influence on the model selection procedure. Then we proceed as follows (see also [155]):

(i) We first split the training data randomly and evenly into M sets with

$$\mathcal{D}_n = \bigcup_{m=1}^{M} I_m \quad \text{with } I_m \subset \mathcal{D}_n \text{ and } |I_m| = |I_{m'}| \tag{4.61}$$

and define

$$\mathcal{D}_n^{(m)} = \bigcup_{k=1,k\neq m}^{M} I_k \tag{4.62}$$

Here, we assume that $2 \leq M \leq n$ and to facilitate the formal depiction $n/M \in \mathbb{N}$.

(ii) For a training configuration (or hyperparameter) λ, we train a model $\hat{f}_{\lambda,\mathcal{D}_n^{(m)}}$. The corresponding validation risk is then

$$\mathcal{R}_{\text{val}}[\hat{f}_{\lambda,\mathcal{D}_n^{(m)}}] = \frac{1}{|I_m|} \sum_{(\mathbf{x},y)\in I_m} c(y, \hat{f}_{\lambda,\mathcal{D}_n^{(m)}}(\mathbf{x})) \tag{4.63}$$

From different hyperparameters, we select the configuration that performs best across all splits

$$\lambda^\star = \arg\min_\lambda \frac{1}{M} \sum_{m=1}^{M} \mathcal{R}_{\text{val}}[\hat{f}_{\lambda,\mathcal{D}_n^{(m)}}] \tag{4.64}$$

This two-step process enables us to determine the performance of a learning process $\hat{f}_{\lambda,\cdot}$ not only on one training dataset, but on many different ones. From a theoretical perspective, a large M is therefore preferable. If $M = n$ we speak of *Leave-one-out cross validation*. A common choice for M is often between 5 and 10 [155].

While M-fold cross-validation offers several advantages over considering a single dataset partition, it does come with higher costs. Specifically, training a model M times is often prohibitively expensive, especially for large deep learning models, making it impractical in many real-world applications.

4.5.4. Robustness and Generalization

We talked in depth about robustness/OOD-generalization[1] in Chapter 2 and Chapter 3. Here, we will mainly discuss what makes the robustness task more challenging compared to the generalization task in the ID setting. Additionally, we point out trade-offs that occur in the robustness task.

While we can theoretically determine the optimal decision rule in classification easily (see the Bayes classifier in Equation 4.49), it is not so straightforward for the OOD-generalization task. The Bayes classifier which achieves the smallest risk in the ID setting, might perform terribly when the environment changes (see for instance our experiment in Subsection 5.5.2). How the optimal classifier looks in the robustness setting depends a lot on the distribution shifts between environments. Depending on the distribution shift, a classifier needs to satisfy certain invariance conditions in order to be robust. We discussed this in detail in Section 3.8.

While it cannot be guaranteed that the Bayes rule will be found in the ID setting[2], a fair model evaluation can be ensured. The law of large numbers ensures that we can estimate the risk of any predictive model. We do not have any comparable results for the robustness task. Since we do not know how a distribution behaves under a distribution shift, there is no way to estimate the risk under the shift. While there are some evaluation schemes (see Subsection 2.2.1), they only give a rough proxy for the risk under distribution shift.

In discussions about robustness, it is crucial to differentiate between extrapolation and interpolation. In this thesis, we consider interpolation as necessary when a new sam-

[1]We use the terms robustness and OOD-generalization interchangeably.

[2]Assumptions on $\mathcal{F}$ and the data generating distribution are necessary

ple $\mathbf{x}$ achieves a positive density under the true density p, expressed as $p(\mathbf{x}) > 0$. Conversely, extrapolation becomes a concern when a new sample $\mathbf{x}$ has 0 density under the training data generating density p, denoted as $p(\mathbf{x}) = 0$. In interpolation, when drawing enough samples from the distribution p, we can obtain samples $\mathbf{x}'$ that are arbitrarily close to $\mathbf{x}$. However, in extrapolation, we cannot expect points that lie in a sufficiently close proximity to $\mathbf{x}$, requiring strong model assumptions to guarantee accurate predictions. Due to characteristics of the specific distribution shift, newly encountered samples might fall outside the support $\text{supp}[p] = \{\mathbf{x} \,|\, p(\mathbf{x}) > 0\}$ of the true density p that describes the training environments. It is therefore of relevance to ask which distribution shift requires extrapolation and which interpolation. For the majority of distribution shifts considered in Section 3.8, novel samples could fall outside the support of the training distribution. Only the prior shift and imbalanced data scenarios are guaranteed to be in the interpolation regime[3]. An overview of these results can be found in Table 4.1.

Distribution Shift	**Interpolation guaranteed**
Covariate Shift	✗
Prior Shift	✓
Imbalanced data	✓
Selection Bias	✗
Source Component shift.	✗
Parent-Child Structure.	✗
Domain Shift	✗
Appearance Shift	✗
Hidden Confounder	✗

Table 4.1. Different distribution shifts and whether extrapolation might be required. For all distribution shifts marked with ✗ there is no guarantee that we are in the interpolation regime. These results can be easily verified by inspecting the graphs that underly the distribution shifts (see Section 3.8).

While we can easily distinguish extrapolation from interpolation in the infinite data regime, it is much more complicated if only a finite number of data samples is given. Consider the normal distribution which has full support (i.e. all inputs elicit positive density), yet, the probability of drawing samples outside a certain range is negligibly low. For instance, drawing one sample from a 1-dimensional normal distribution $\mathcal{N}(\mu, \sigma)$ every day implies an event that is 8σ away from its mean would occur approximately every 2.2 trillion years, which practically means it has never occurred in the history of the universe [169]. In theory, samples far from the mean would require interpolation, but in practical application, it tends to align more with extrapolation. Furthermore, the curse of dimensionality (see Section 4.5) implies that in high-dimensional spaces, samples in close proximity to an input are very unlikely to be drawn. This might render extrapolation and interpolation from a practical standpoint equivalent in high dimensions. Understanding the relation between extrapolation and interpolation in the finite data regime, specifically with respect to distribution shifts is a challenging and exciting avenue for future research.

As we discussed various trade-offs in the ID setting, it is important to note that trade-offs also arise in the context of robustness. One notorious trade-off is between fitting the ID data particularly well and finding an invariance that might involve discarding features that are predictive in the ID setting. We delve into this trade-off in Chapter 5. Related

[3]"Only the mass of the distribution shifts, but the support is not left"

to this trade-off, a risk decomposition was proposed in [33] that is adapted here to our framework. Let $\mathcal{R}^{\text{seen}}[f]$ be the risk of a function $f \in \mathcal{F}$ in the seen training environments, and $f^{e,\star} = \arg\min_{f \in \mathcal{F}} \mathcal{R}^e[f]$ representing the function that minimizes risk within a novel environment e. We can decompose the *total bias* into a *transfer bias* and an *incomplete information error*:

$$\underbrace{\mathcal{R}^e[\hat{f}_{\lambda,\mathcal{D}^{\text{seen}}}] - \mathcal{R}^e[f^{e,\star}]}_{\text{total bias}} = \underbrace{\mathcal{R}^e[\hat{f}_{\lambda,\mathcal{D}^{\text{seen}}}] - \mathcal{R}^e[\hat{f}_{\lambda,\mathcal{D}^e}]}_{\text{transfer bias}} + \underbrace{\mathcal{R}^e[\hat{f}_{\lambda,\mathcal{D}^e}] - \mathcal{R}^e[f^{e,\star}]}_{\text{incomplete information error}} \tag{4.65}$$

where e is a novel environment. The total bias represents the discrepancy arising from using the function $\hat{f}_{\lambda,\mathcal{D}^{\text{seen}}}$ trained on the training dataset $\mathcal{D}^{\text{seen}}$ with specific training parameters λ (e.g., an invariance objective as proposed in Chapter 5) rather than the optimal function $f^{e,\star}$ in $\mathcal{F}$. Conversely, the transfer bias characterizes the deviation resulting from training on $\mathcal{D}^{\text{seen}}$ instead of samples $\mathcal{D}^e$ drawn from the novel environment. If the predictive model achieves equal results in all environments (i.e. it is robust), the transfer bias should be close to 0. The incomplete information error denotes the error attributed to idiosyncrasies of the learning procedure $\hat{f}_{\lambda,\mathcal{D}^e}$ applied on data from the novel environment $\mathcal{D}^e$ instead of using the optimal function $f^{e,\star} \in \mathcal{F}$. For instance, a learning procedure might disregard certain features to fulfill an invariance objective.

4.6. Generative Models

Many machine learning models aim to estimate the density of a random variable $\mathbf{X}$ or to sample from its distribution $P_{\mathbf{X}}$. Various models targeting these tasks, including Variational Autoencoder (VAEs) [170], generative adversarial network (GANs) [171] and the recently successful diffusion models [172–174], exist. In this section, we only discuss two models since they bear greater relevance to this thesis.

Autoregressive Models An interesting model class we briefly like to mention is the class of autoregressive models [175] which offer a particular causal interpretation. They model each factor in the factorization of the joint distribution separately by evoking the chain rule of probability

$$P(X_1, \dots, X_D) = \prod_{i=1}^{D} P(X_i \mid X_1, \dots, X_{i-1}) \tag{4.66}$$

If the ordering of the variables conforms to a causal ordering, i.e. nodes X_i with $i < j$ cannot be descendants of X_j, then the autoregressive model corresponds to the causal factorization. This results from the observation that any set of nodes $\mathbf{A} \subset \{X_1, \dots, X_{i-1}\}$ which is disjoint from $\mathbf{X}_{pa(i)}$ satisfies the d-separation statement $X_i \perp_d \mathbf{A} \mid \mathbf{X}_{pa(i)}$. With the causal Markov condition, we then obtain that $P(X_i \mid X_1, \dots, X_{i-1}) = P(X_i \mid \mathbf{X}_{pa(i)})$. Hence, an autoregressive model on a causal ordering is a causal model.

Normalizing Flows The following normalizing flow part is adapted from our work in [31]. Normalizing flows model complex distributions by means of invertible functions T (chosen from some model space $\mathcal{T}$), which map the densities of interest to latent normal distributions. Normalizing flows are typically built with specialized neural networks that are invertible by construction and have tractable Jacobian determinants. They are used for density estimation and sampling of a target density (for an overview see [176]). This in turn allows optimizing information theoretic objectives in a convenient and mathematically sound way.

Together with a reference distribution p_{ref}, a normalizing flow T defines a new distribution $\nu_T = (T(\mathbf{x}))_{\#}^{-1} p_{\text{ref}}$ which is called the push-forward of the reference distribution p_{ref} [177]. By drawing samples from p_{ref} and applying T on these samples we obtain samples from this new distribution. The density of this so-obtained distribution p_{ν_T} can be derived from the change of variables formula:

$$p_{\nu_T}(\mathbf{x}) = p_{\text{ref}}(T(\mathbf{x}))|\nabla_{\mathbf{x}} T(\mathbf{x})| \tag{4.67}$$

A normalizing flow T or T_θ is usually represented by a neural network architecture with parameters θ. In this case, we denote the corresponding density as $p(\mathbf{x};\theta) = p_{\nu_{T_\theta}}(\mathbf{x})$.

We can compute the KL-Divergence between the true density $p_\star$ and the modeled density [178]

$$\begin{aligned}
\mathcal{L}(\theta) &= D_{\text{KL}}(p_\star \| p(\cdot;\theta)) && (4.68)\\
&= -\mathbb{E}_{\mathbf{X}\sim p_\star}[\log p(\mathbf{X};\theta)] + \mathbb{E}_{\mathbf{X}\sim p}[\log p_\star(\mathbf{X}))] && (4.69)\\
&= -\mathbb{E}_{\mathbf{X}\sim p_\star}[\log p_{\text{ref}}(T^{-1}(\mathbf{X},\theta);\psi) + \log|\det J_{T_\theta^{-1}}(\mathbf{X};\theta)|] + \text{const} && (4.70)
\end{aligned}$$

These equations also show how we can obtain $p_\star$: we need to minimize $\mathcal{L}(\theta)$. A typical choice for reference distribution is the isotropic normal distribution (i.e. normal distribution with the identity matrix as covariance matrix) in which case we get

$$\begin{aligned}
\mathcal{L}(\theta) &= D_{\text{KL}}(p_\star(\mathbf{x}) \| p(\mathbf{x};\theta)) && (4.71)\\
&= \mathbb{E}_{\mathbf{X}}\Big[\|T(\mathbf{X})\|^2/2 - \log|\det \nabla_{\mathbf{x}} T(\mathbf{X})|\Big] + \text{const} && (4.72)
\end{aligned}$$

If we were to model a conditional distribution $p(y \mid \mathbf{x})$, the change of variables formula gives

$$p_{\nu_T}(y \mid \mathbf{x}) = p_{\text{ref}}(T(y;\mathbf{x}))|\nabla_y T(y;\mathbf{x})| \tag{4.73}$$

In this case, the KL Divergence takes the following form

$$\begin{aligned}
&\mathbb{E}_{\mathbf{X}}[D_{\text{KL}}(p_{Y|\mathbf{X}} \| p_{\nu_T})]\\
=&\mathbb{E}_{\mathbf{X}}\left[\mathbb{E}_{Y|\mathbf{X}}\left[\log\left(\frac{p_{Y|\mathbf{X}}}{p_{\nu_T}}\right)\right]\right]\\
=&-H(Y \mid \mathbf{X}) - \mathbb{E}_{\mathbf{X},Y}[\log p_{\nu_T}(Y \mid \mathbf{X})]\\
=&-H(Y \mid \mathbf{X}) + \mathbb{E}_{\mathbf{X},Y}[-\log p_{\text{ref}}(T(y;\mathbf{x})\\
&-\log|\nabla_y T(y;\mathbf{x})|] && (4.74)
\end{aligned}$$

The last two terms in Equation 4.74, namely

$$\mathbb{E}_{\mathbf{X},Y}[-\log p_{\text{ref}}(T(Y;\mathbf{X}) - \log|\nabla_y T(Y;\mathbf{X})|]$$

correspond to the negative log-likelihood (NLL) for conditional flows with reference distribution p_{ref} in latent space. If the reference distribution is assumed to be standard normal, the NLL is given as

$$\mathcal{L}_{\text{NLL}}(T) := \mathbb{E}_{\mathbf{X},Y}\Big[\|T(Y;\mathbf{X}\|^2/2 - \log|\det\nabla_y T(Y;\mathbf{X})|\Big] + \text{const} \tag{4.75}$$

In Remark 10 we will show that normalizing flows generalize Structural Causal Models (SCM) that employ the additive noise assumption.

Learning Robust Models using the Principle of ICM 5

The content of this chapter is a direct adaptation from our work in [31].

Abstract Standard supervised learning breaks down under data distribution shift. However, the principle of independent causal mechanisms (ICM, [32]) can turn this weakness into an opportunity: one can take advantage of distribution shift between different environments during training in order to obtain more robust models. We propose a new gradient-based learning framework whose objective function is derived from the ICM principle. We show theoretically and experimentally that neural networks trained in this framework focus on relations remaining invariant across environments and ignore unstable ones. Moreover, we prove that the recovered stable relations correspond to the true causal mechanisms under certain conditions, turning domain generalization into a causal discovery problem. In both regression and classification, the resulting models generalize well to unseen scenarios where traditionally trained models fail.

5.1. Introduction

Standard supervised learning has shown impressive results when training and test samples follow the same distribution. However, many real-world applications do not conform to this setting, so that research successes do not readily translate into practice (see Subsection 2.2.3 or [179]). *Domain Generalization* (DG) addresses this problem: it aims at training models that generalize well under domain shift. In contrast to Domain *Adaption*, where a few labeled and/or many unlabeled examples are provided for each target test domain, in DG absolutely no data is available from the test domains' distributions making the problem unsolvable in general. For a more thorough introduction to DG and related problems see Section 2.2.

In this chapter, we view the problem of DG specifically using ideas from causal discovery. This viewpoint makes the problem of DG well-posed: we assume that there exists a feature vector $h^\star(\mathbf{X})$ whose relation to the target variable Y is invariant across all environments. Consequently, the conditional probability $p(Y \mid h^\star(\mathbf{X}))$ has predictive power in each environment. From a causal perspective, changes between domains or environments can be described as interventions; and causal relationships – unlike purely statistical ones – remain invariant across environments unless explicitly changed under intervention. This is due to the fundamental principle of "Independent Causal Mechanisms" which we introduced in Section 3.6. From a causal standpoint, finding robust models is therefore a *causal* *discovery* task (see also Section 3.8 or [7, 180]). Taking a causal perspective on DG, we aim at identifying features which (i) have an invariant relationship to the target variable Y and (ii) are maximally informative about Y. This problem has already been addressed with some simplifying assumptions and a discrete combinatorial search by [33, 72], but we make weaker assumptions and enable gradient-based optimization. The latter is attractive because it readily scales to high dimensions and offers the possibility to *learn* very informative features, instead of merely selecting among predefined ones. Approaches to invariant relations similar to ours were taken by [181], who restrict themselves to linear relations, and [34, 126], who consider a weaker notion of invariance. Problems (i) and (ii) are quite intricate because the search space has combinatorial complexity and testing for conditional independence in high dimensions is notoriously difficult. Our main contributions to this problem are the following: *First*, by connecting invariant (causal) relations with normalizing flows, we propose a differentiable two-part objective of the form $I(Y; h(\mathbf{X})) + \lambda_I \mathcal{L}_I$, where I is the mutual information and $\mathcal{L}_I$ enforces the invariance of the relation between $h(\mathbf{X})$ and Y across all environments. This objective operationalizes the ICM principle with a trade-off between feature informativeness and invariance controlled by parameter λ_I. Our formulation generalizes existing work because our objective is not restricted to linear models. *Second*, we take advantage of the continuous objective in three important ways:

(1) We can learn invariant new features, whereas graph-based methods as in e.g. [33] can only select features from a pre-defined set.

(2) Our approach does not suffer from the scalability problems of combinatorial optimization methods as proposed in e.g. [29] and [72].

(3) Our optimization via normalizing flows, i.e. in the form of a density estimation task, facilitates accurate maximization of the mutual information.

Third, we show how our objective simplifies in important special cases and under which conditions its optimal solution identifies the true causal parents of the target variable Y. We empirically demonstrate that the new method achieves good results on two datasets proposed in the literature.

5.2. Related Work

Different types of invariances have been considered in the field of DG. We introduced them in detail in Section 2.4 and discuss here the important ones in relation to the work presented in this chapter. One type is defined on the feature level, i.e. features $h(\mathbf{X})$ are invariant across environments if they follow the same distribution in all environments (e.g. [81, 182, 183]). However, this form of invariance is problematic since the distribution of the target variable might change between environments, which should induce a corresponding change in the distribution of $h(\mathbf{X})$ (see for instance Section 2.4). A more plausible and theoretically justified assumption is the invariance of *relations* [29, 33, 72]. The relation between a target Y and features $h(\mathbf{X})$ is invariant across environments, if the conditional distribution $p(Y \mid h(\mathbf{X}))$ remains unchanged in all environments. This is what we termed in Section 2.4 the *causal invariance*. Existing approaches exhaustively model conditional distributions for all possible feature selections and check for the invariance property [29, 33, 72], which scales poorly for large feature spaces. We derive a theoretical result connecting *normalizing flows* and *invariant relations*, which enables gradient-based learning of an invariant solution. In order to exploit our formulation, we also use the Hilbert-Schmidt-Independence Criterion that has been used for robust learning by [184] in the one environment setting. [34, 126, 185] also propose gradient-based learning frameworks, which exploit a weaker notion of invariance: They aim to match the conditional expectations across environments, whereas we address the harder problem of matching the entire conditional distributions. The connection between DG, invariance and causality has been pointed out for instance by [7, 72, 186] and we discussed it systematically in Section 3.8. From a causal perspective, DG is a causal discovery task [7]. For studies on causal discovery in the purely observational setting see e.g., [94, 187, 188], but they do not take advantage of variations across environments. The case of different environments has been studied by [29, 33, 93, 180, 189–192]. Most of these approaches rely on combinatorial optimization or are restricted to linear mechanisms, whereas our continuous objective efficiently optimizes very general non-linear models. The distinctive property of causal relations to remain invariant across environments in the absence of direct interventions has been known since at least the 1930s [193, 194]. However, its crucial role as a tool for causal discovery was – to the best of our knowledge– only recently recognized by [29]. Their estimator – *Invariant Causal Prediction* (ICP) – returns the intersection of all subsets of variables that have an invariant relation with respect to Y. The output is shown to be the set of the direct causes of Y under suitable conditions. We considered ICP in more detail in Section 3.9. Again, this approach requires linear models and an exhaustive search over all possible variable sets $\mathbf{X}_S$. Extensions to time series and non-linear additive noise models were studied in [73, 195]. Our treatment of invariance is inspired by these papers and also discusses identifiability results, i.e. conditions when the identified variables are indeed the direct causes, with two key differences: Firstly, we propose a formulation that allows for a gradient-based learning and does not need strong assumptions on the underlying causal model. Second, while ICP tends to exclude features from the parent set when in doubt, our algorithm prefers to err toward best predictive performance in this situation.

5.3. Preliminaries

In the following, we introduce the basics of this work as well as the connection between DG and causality. Basics on causality are presented in Chapter 3. We first define our notation as follows: We denote the set of all variables describing the system under study as $\widetilde{\mathbf{X}} = \{X_1, \dots, X_D\}$. One of these variables will be singled out as our prediction target, whereas the remaining ones are observed and may serve as predictors. To clarify notation, we call the target variable $Y \equiv X_i$ for some $i \in \{1, \dots, D\}$, and the remaining observations are $\mathbf{X} = \widetilde{\mathbf{X}} \setminus \{Y\}$. In the Domain Generalization context, we employ the same notation as in Section 2.2. Here, symbols with superscript, e.g. Y^e, also refer to a specific environment, whereas symbols without refer to data pooled over all environments. Similar to Remark 4 (or [93]) we consider the environment to be an RV E and therefore a system variable. This gives an additional view on causal discovery and the DG problem.

5.3.1. Invariance and the Principle of ICM

DG is in general unsolvable because distributions between seen and unseen environments could differ arbitrarily. In order to transfer knowledge from $\mathcal{E}_{\text{seen}}$ to $\mathcal{E}_{\text{unseen}}$, we have to make assumptions on how seen and unseen environments relate. These assumptions have a close link to causality as elaborated in Section 3.8. We assume certain relations between variables remain invariant across all environments. A subset $\mathbf{X}_S \subset \mathbf{X}$ of variables *elicits an invariant relation* or *satisfies the invariance property* with respect to Y over a subset $W \subset \mathcal{E}$ of environments if

$$\forall e, e' \in W\colon \quad P(Y^e \,|\, \mathbf{X}_S^e = u) = P(Y^{e'} \,|\, \mathbf{X}_S^{e'} = u) \tag{5.1}$$

for all u where both conditional distributions are well-defined. It can be equivalently defined by $Y \perp E \,|\, \mathbf{X}_S$ and $I(Y; E \,|\, \mathbf{X}_S) = 0$ for E restricted to W. The *invariance property* for computed features $h(\mathbf{X})$ is defined analogously by the relation $Y \perp E \,|\, h(\mathbf{X})$. This corresponds to the causal invariance in Definition 1. Although we can only test for Equation 5.1 in $\mathcal{E}_{\text{seen}}$, taking a causal perspective allows us to derive plausible conditions for an invariance to remain valid in all environments $\mathcal{E}$. In brief, we assume that environments correspond to interventions in the system and invariance arises from the principle of *independent causal mechanisms* (see Section 3.6). We specify these conditions later in Assumption 1 and 2. At first, consider the joint density $p_{\widetilde{\mathbf{X}}}(\widetilde{\mathbf{X}})$. The chain rule offers a combinatorial number of ways to decompose this distribution into a product of conditionals. Among those, the *causal factorization*

$$p_{\widetilde{\mathbf{X}}}(x_1, \ldots, x_D) = \textstyle\prod_{i=1}^{D} p_i(x_i \,|\, \mathbf{x}_{pa(i)}) \tag{5.2}$$

is singled out by conditioning each X_i onto its *direct causes* or *causal parents* $\mathbf{X}_{pa(i)}$, where $pa(i)$ denotes the appropriate index set. The special properties of this factorization are discussed in Section 3.6. The conditionals p_i of the causal factorization are called *causal mechanisms*. An *intervention* onto the system is defined by replacing one or several factors in the decomposition with different (conditional) densities $\overline{p}$ (for interventions in the context of SCMs see Subsection 3.3.2). Here, we distinguish *soft-interventions* where $\overline{p}_j(x_j \,|\, \mathbf{x}_{pa(j)}) \neq p_j(x_j \,|\, \mathbf{x}_{pa(j)})$ and *hard-interventions* where $\overline{p}_j(x_j \,|\, \mathbf{x}_{pa(j)}) = \overline{p}_j(x_j)$ is a density which does not depend on $x_{pa(j)}$ (e.g. an atomic intervention where x_j is set to a specific value $\overline{x}$). The resulting joint distribution for a single intervention is

$$\overline{p}_{\widetilde{\mathbf{X}}}(x_1, \ldots, x_D) = \overline{p}_j(x_j \,|\, \mathbf{x}_{pa(j)}) \textstyle\prod_{i=1, i\neq j}^{D} p_i(x_i \,|\, \mathbf{x}_{pa(i)}) \tag{5.3}$$

and extends to multiple simultaneous interventions in an obvious way. The principle of *independent causal mechanisms* (ICM) states that every mechanism acts independently of the others (see Section 3.6). Consequently, an intervention replacing p_j with $\overline{p}_j$ has no effect on the other factors $p_{i\neq j}$, as indicated by Equation 5.3. This is a crucial property of the causal decomposition – alternative factorizations do not exhibit this behavior. Instead, a coordinated modification of several factors is generally required to model the effect of an intervention in a non-causal decomposition. We utilize this principle as a tool to train *robust* models. To do so, we make two additional assumptions, similar to [29] and [195]:

Assumption. *We pose the following two assumptions*

(1) *Any differences in the joint distributions $p^e_{\widetilde{\mathbf{X}}}$ from one environment to the other are fully explainable as interventions: replacing factors $p_i^e(x_i \,|\, \mathbf{x}_{pa(i)})$ in environment e with factors $p_i^{e'}(x_i \,|\, \mathbf{x}_{pa(i)})$ in environment e' (for some subset of the variables) is the only admissible change.*

(2) *The mechanism $p(y \,|\, \mathbf{x}_{pa(Y)})$ for the target variable Y is invariant under changes of environment, i.e. we require conditional independence $Y \perp E \,|\, \mathbf{X}_{pa(Y)}$.*

Assumption 2 implies that Y must not directly depend on E. Consequences in the case of omitted variables are discussed in the following remark.

Remark 7 (Causal Sufficiency and Omitted Variables)**.** *It has important consequences when there exist omitted variables* $\mathbf{W}$*, which affect* Y *but have not been measured. This is specifically problematic in two scenarios. First, if there does not exist a set of variables that* d*-separate* Y *from* $\mathbf{W}$ *and there does also not exist a set of variables that* d*-separate* E *from* $\mathbf{W}$*. In this case,* Y *and* E *are no longer* d*-separated by* $\mathbf{X}_{pa(Y)}$ *and Assumption 2 is in general unsatisfiable. Second, if* $\mathbf{W}$ *contains a hidden confounder affecting both* $\mathbf{X}_{pa(Y)}$ *and* Y *(causal sufficiency is violated) and* E *directly affects* $\mathbf{X}_{pa(Y)}$*, i.e.* $E \rightarrow \mathbf{X}_{pa(Y)}$*. In this case,* $\mathbf{X}_{pa(Y)}$ *can become a collider (with the graphical structure* $E \rightarrow \mathbf{X}_{pa(Y)} \leftarrow \mathbf{W} \rightarrow Y$*) and therefore* Y *and* E *are no longer* d*-separated by* $\mathbf{X}_{pa(Y)}$*. Consequentially, Assumption 2 is unsatisfiable in general. In both scenarios, the method we propose in this chapter is unable to find an invariant mechanism.*

If we knew the causal decomposition, we could use these assumptions directly to train a robust model for Y – we would simply regress Y on its parents $\mathbf{X}_{pa(Y)}$. However, we only require that a causal decomposition with these properties exists, but do not assume that it is known. Instead, our method uses the assumptions indirectly – by simultaneously considering data from different environments – to identify a stable regressor for Y. We call a regressor stable if it solely relies on predictors whose relationship to Y remains invariant across environments, i.e. is not influenced by any intervention. By assumption 2, such a regressor always exists. However, predictor variables beyond $\mathbf{X}_{pa(Y)}$, e.g. children of Y or parents of children, may be included into our model as long as their relationship to Y remains invariant across all environments. We discuss this in the following Remark

Remark 8 (Using Causal Effects for Prediction)**.** *The estimator we propose in this chapter might use predictor variables beyond* $\mathbf{X}_{pa(Y)}$ *as well, e.g., children of* Y *or parents of children, provided their relationships to* Y *do not depend on the environment. The case of children is especially interesting: Suppose* X_j *is a noisy measurement of* Y*, described by the causal mechanism* $p(x_j \mid y)$*. As long as the measurement device works identically in all environments, including* X_j *as a predictor of* Y *is desirable, despite it being a child.*

In general, prediction accuracy will be maximized when all suitable predictor variables are included into the model. Accordingly, our algorithm will asymptotically identify the full set of stable predictors for Y. In addition, we will prove under which conditions this set contains exactly the parents of Y. The following example shows the special role of the parents of Y for robustness.

Example 6. Suppose we would like to estimate the gas consumption of a car. In a sufficiently narrow setting, the total amount of money spent on gas might be a simple and accurate predictor. However, gas prices vary dramatically between countries and over time, so statistical models relying on it will not be robust, even if they fit the training data very well. Gas costs are an *effect* of gas consumption, and this relationship is unstable due to external influences, such as varying tax policies across different countries. In contrast, predictions based on the *causes* of gas consumption (e.g., car model, geography, local speed limits, and owner's driving habits) tend to be much more robust, because these causal relations are intrinsic to the system and not subjected to external influences. See Figure 5.1 for a simplified illustration of this scenario. Note that there is a trade-off here: Including gas costs in the

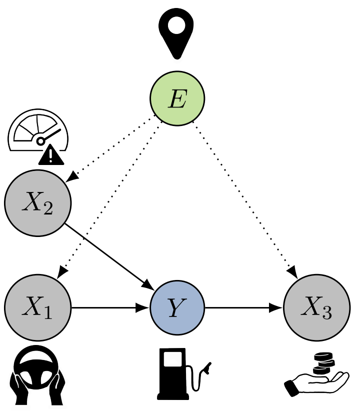


Figure 5.1. Simplifeid illustration of Example 6. The country (or tax policies) E represents the environment variable. The task asks to predict gas consumption Y from the local speed limits X_2, the owner's driving habits X_1, and X_3 the paid gas costs. Only X_2 and X_1 elicit a stable relation to Y and promise robustness to predict Y.

model will improve estimation accuracy when gas prices remain sufficiently stable, but will impair results otherwise. By considering the same phenomenon in several environments simultaneously, we hope to gain enough information to adjust this trade-off properly.

In the gas example, countries can be considered as environments that "intervene" on the relationship between consumed gas and money spent, e.g., by applying different tax policies. In contrast, interventions changing the impact of motor properties or geography on gas consumption are much less plausible – powerful motors and steep roads will always lead to higher consumption.

5.3.2. Domain Generalization

To exploit the principle of ICM for DG, we formulate the DG problem in Equation 2.3 in an information theoretical context as follows

$$h^\star := \underset{h\in\mathcal{H}}{\arg\max}\left\{\min_{e\in\mathcal{E}} I(Y^e; h(\mathbf{X}^e))\right\} \qquad \text{s.t.} \quad Y \perp E \mid h(\mathbf{X}) \tag{5.4}$$

The optimization problem in Equation 5.4 asks to find features $h(\mathbf{X})$ which are maximally informative in the worst environment subject to the invariance constraint. where $h \in \mathcal{H}$ denotes a learnable feature extraction function $h\colon \mathbb{R}^D \to \mathbb{R}^M$ where M is a hyperparameter. This optimization problem defines a maximin objective: The features $h(\mathbf{X})$ should be as informative as possible about the response Y even in the most difficult environment, while conforming to the ICM constraint that the relationship between features and response must remain invariant across all environments. In principle, our approach can also optimize related objectives like the average mutual information over environments. However, very good performance in a majority of the environments could then mask failure in a single (outlier) environment. We opted for the maximin formulation to avoid this. On the other hand there might be scenarios where the maxmin formulation is limited. For instance when the training signal is very noisy in one environment, the classifier might discard valuable information from the other environments. As it stands, Equation 5.4 is hard to optimize, because traditional independence tests for the constraint $Y \perp E \mid h(\mathbf{X})$ cannot cope with conditioning variables selected from a potentially infinitely large space

$\mathcal{H}$. A re-formulation of the DG problem to circumvent these issues is our main theoretical contribution.

5.3.3. Normalizing Flows

We introduced normalizing flows in Section 4.6 as generative models that allow us to estimate densities and sample from some learned target distribution. Here, we also represent the conditional distribution $P(Y \mid h(\mathbf{X}))$ by a *conditional* normalizing flow (see e.g., [196]). To optimize a conditional normalizing flow, we aim to minimize the negative log-likelihood (NLL) loss of Y under T, given by

$$\mathcal{L}_{\text{NLL}}(T, h) := \mathbb{E}_{h(\mathbf{X}),Y}\Big[\|T(Y; h(\mathbf{X})\|^2/2 - \log|\det \nabla_y T(Y; h(\mathbf{X}))|\Big] + \text{const} \tag{5.5}$$

where $\det \nabla_y T$ is the Jacobian determinant and const $= \dim(Y)\log(\sqrt{2\pi})$ is a constant that can be dropped [176]. Equation 5.5 can be derived from the change of variables formula and the assumption that T maps to a standard normal distribution (see Section 4.6). If we consider the NLL on a particular environment $e \in \mathcal{E}$, we denote this with $\mathcal{L}^e_{\text{NLL}}$. Lemma 1 shows that normalizing flows optimized by NLL are indeed applicable to our problem:

Lemma 1. *Let*

$$h^\star, T^\star := \arg\min_{h\in\mathcal{H}, T\in\mathcal{T}} \mathcal{L}_{\text{NLL}}(T, h) \tag{5.6}$$

be the solution of the NLL minimization problem on a sufficiently rich function space $\mathcal{T}$, i.e. we assume that for all $h \in \mathcal{H}$ there exists one $T \in \mathcal{T}$ with $\mathbb{E}_{h(\mathbf{X})}[D_{KL}(p_{Y|h(\mathbf{X})} \| p_{\nu_T})] = 0$. Then the following properties hold for any set $\mathcal{H}$ of feature extractors:

(a) $h^\star$ also maximizes the mutual information, i.e. $h^\star = \arg\max_{g\in\mathcal{H}} I(g(\mathbf{X}); Y)$

(b) $h^\star$ and the latent variables $R = T^\star(Y; h^\star(\mathbf{X}))$ are independent: $h^\star(\mathbf{X}) \perp R$

Proof. For an introduction to normalizing flows and the notation used in this proof see Section 4.6. We first show statement (a). From Equation 4.74, we obtain

$$-\mathbb{E}_{h(\mathbf{X}),Y}[\log p_{\nu_T}(Y \mid h(\mathbf{X}))] \geq H(Y \mid h(\mathbf{X})) \tag{5.7}$$

for all $h \in \mathcal{H}, T \in \mathcal{T}$. We furthermore have

$$\min_{T\in\mathcal{T}} -\mathbb{E}_{h(\mathbf{X}),Y}[\log p_{\nu_T}(Y \mid h(\mathbf{X}))] = H(Y \mid h(\mathbf{X})) \tag{5.8}$$

due to our expressiveness assumptions on $\mathcal{T}$. Therefore, we obtain

$$\min_{h\in\mathcal{H}, T\in\mathcal{T}} -\mathbb{E}_{h(\mathbf{X}),Y}[\log p_{\nu_T}(Y \mid h(\mathbf{X}))] = \min_{h\in\mathcal{H}} H(Y \mid h(\mathbf{X})) \tag{5.9}$$

Since we have $I(Y; h(\mathbf{X})) = H(Y) - H(Y \mid h(\mathbf{X}))$ and only the second term depends on h, we can conclude statement (a).

Now, we prove statement (b). For convenience, we denote $T(Y; h(\mathbf{X})) = R$ and $h(\mathbf{X}) = Z$. Due to the expressiveness of $\mathcal{T}$, we achieve

$$\mathbb{E}_Z[D_{\mathrm{KL}}(p_{Y|Z} \| p_{\nu_{T^\star}})] = 0 \tag{5.10}$$

and therefore $p_{Y|Z}(y \mid z) = p_{\nu_{T^\star}} = p_{\mathrm{ref}}(T(y; z))|\nabla_y T^{-1}(y; z)|$. By applying the change of variables formula two times, we get

$$\begin{aligned} p_{R|Z}(r \mid z) &= p_{Y|Z}(T^{-1}(r; z)|z) \cdot |\nabla_y T^{-1}(r; z)| \\ &= p_{\mathrm{ref}}(T(T^{-1}(r; z); z)) \cdot |\nabla_y T(y; z)| \\ &\quad \cdot |\nabla_y T^{-1}(r; z)| \\ &= p_{\mathrm{ref}}(r) \cdot 1 \end{aligned}$$

Since the density p_{ref} is independent of Z, we obtain $R \perp Z$ which concludes the proof of (b). □

Statement (a) guarantees that $h^\star$ extracts as much information about Y as possible. Hence, the main objective in Equation 5.4 becomes equivalent to optimizing Equation 5.5 when we restrict the space $\mathcal{H}$ of admissible feature extractors to the subspace $\mathcal{H}_\perp$ satisfying the invariance constraint $Y \perp E \mid h(\mathbf{X})$:

$$\operatorname*{arg\,min}_{h \in \mathcal{H}_\perp} \min_{T \in \mathcal{T}} \max_{e \in \mathcal{E}} \mathcal{L}^e_{\mathrm{NLL}}(T; h) = \operatorname*{arg\,max}_{h \in \mathcal{H}_\perp} \min_{e \in \mathcal{E}} I(Y^e; h(\mathbf{X}^e)) \tag{5.11}$$

We give a sketch of the proof of Equation 5.11 in the following remark.

Remark 9 (Proof of Equation 5.11)**.** *Equation 5.11 can be concluded from Lemma 1 and its assumptions. Let $h \in \mathcal{H}_\perp$ be a feature extractor that satisfies $Y \perp E \mid h(\mathbf{X})$. Then, it is easily seen that there exists a $T^\star \in \mathcal{T}$ with*

$$\mathcal{L}_{\mathrm{NLL}}(T^\star; h) = \min_{T \in \mathcal{T}} \mathcal{L}_{\mathrm{NLL}}(T, h) \tag{5.12}$$

Furthermore, for each environment e there exists a $T^\star_e \in \mathcal{T}$ with

$$\mathcal{L}^e_{\mathrm{NLL}}(T^\star_e, h) = \min_{T \in \mathcal{T}} \mathcal{L}^e_{\mathrm{NLL}}(T, h) \tag{5.13}$$

for all $e \in \mathcal{E}$. Since the conditional densities $p(y \mid h(\mathbf{X}))$ are invariant across all environments, we have

$$H(Y^e \mid h(\mathbf{X}^e)) = \mathcal{L}^e_{\mathrm{NLL}}(T^\star_e; h) = \mathcal{L}^e_{\mathrm{NLL}}(T^\star; h) \tag{5.14}$$

for all $e \in \mathcal{E}$. Therefore,

$$\begin{aligned} \operatorname*{arg\,min}_{h \in \mathcal{H}_\perp} \min_{T \in \mathcal{T}} \max_{e \in \mathcal{E}} \mathcal{L}^e_{\mathrm{NLL}}(T; h) &= \operatorname*{arg\,min}_{h \in \mathcal{H}_\perp} \max_{e \in \mathcal{E}} \mathcal{L}^e_{\mathrm{NLL}}(T^\star; h) && (5.15) \\ &= \operatorname*{arg\,max}_{h \in \mathcal{H}_\perp} \max_{e \in \mathcal{E}} H(Y^e \mid h(\mathbf{X}^e)) && (5.16) \\ &= \operatorname*{arg\,max}_{h \in \mathcal{H}_\perp} \min_{e \in \mathcal{E}} I(Y^e; h(\mathbf{X}^e)) && (5.17) \end{aligned}$$

The last equation follows from the relation $I(Y^e; h(\mathbf{X}^e)) = H(Y^e) - H(Y^e \mid h(\mathbf{X}^e))$.

Statement (b) in Lemma 1 ensures that the flow indeed implements a valid structural equation, which requires that R can be sampled independently of the features $h(\mathbf{X})$. This fact generalizes the additive noise assumption that is common to a large portion of the Structural Causal Models literature as discussed in the following Remark.

Remark 10 (Normalizing Flows and Additive Noise Models)**.** *The additive noise assumption in SCMs implies that structural assignments are of the form $Y = f(\mathbf{X}_S) + R$ where R is the noise variable that is independent of the variable selection $\mathbf{X}_S$. Hence, we can also compute the noise/residual via $R = Y - f(\mathbf{X}_S)$. This computation represents a diffeomorphism of the form $T_f(Y;\mathbf{X}_S) = Y - f(\mathbf{X}_S)$.*

We represent the conditional distribution $P(Y \mid \mathbf{X}_S)$ using a conditional *normalizing flow (see e.g. Section 4.6 or [196]). In our work, we seek a mapping $R = T(Y;\mathbf{X}_S)$ that is diffeomorphic in Y such that $R \sim \mathcal{N}(0,1) \perp h(\mathbf{X})$. The inverse $Y = F(R;\mathbf{X}_S)$ takes the role of a structural equation for the mechanism $p(Y \mid \mathbf{X}_S)$ with R being the corresponding noise variable. Therefore, we can view this approach as a generalization of the well-studied additive Gaussian noise model*[1].

5.3.4. Hilbert Schmidt Independence Criterion (HSIC)

The Hilbert-Schmidt Independence Criterion (HSIC) is a kernel-based measure for independence which is in expectation 0 if and only if the compared random variables are independent [198]. An empirical estimate of HSIC(A, B) for two random variables A, B is given by

$$\widehat{\text{HSIC}}(\{a_j\}_{j=1}^n, \{b_j\}_{j=1}^n) = \frac{1}{(n-1)^2}\,\mathrm{tr}(LKL'K) \tag{5.18}$$

where tr is the trace operator. $L_{ij} = l(a^i, a^j)$ and $L'_{ij} = l'(b^i, b^j)$ are kernel matrices for given kernels l and l'. The matrix K is a centering matrix $K_{i,j} = \delta_{i,j} - 1/n$ where $\delta_{i,j}$ is the Kronecker delta that is 1 if $i = j$ and 0 otherwise. $a_1, \dots, a_n$ and $b_1, \dots, b_n$ are independent realizations of the RVs A and B. For more details on HSIC as well as a theoretical derivation see [198].

5.4. Method: Learning Invariances using the Principle of ICM

In the following, we propose a way of indirectly expressing the constraint in Equation 5.4 via normalizing flows. Thereafter, we combine this result with Lemma 1 to obtain a differentiable objective for solving the DG problem. We also present important simplifications for least squares regression and softmax classification and discuss the relations of our approach with causal discovery.

[1] F is the concatenation of the normal CDF with the inverse CDF of $P(Y \mid \mathbf{X}_S)$, see [197].

5.4.1. Learning the Invariance Property

The following theorem establishes a connection between invariant relations, prediction residuals and normalizing flows. The key consequence is that a suitably trained normalizing flow translates the statistical independence of the latent variable R from the features and environment $(h(\mathbf{X}), E)$ into the desired invariance of the mechanism $P(Y \mid h(\mathbf{X}))$ under changes of E. We will exploit this for an elegant reformulation of the DG problem (Equation 5.4) into the objective (Equation 5.20) below.

Theorem 1. *Let h be a differentiable function and $Y, \mathbf{X}, E$ be RVs. Furthermore, let $R = T(Y; h(\mathbf{X}))$ be a continuous, differentiable function that is a diffeomorphism in Y. Suppose that $R \perp (h(\mathbf{X}), E)$. Then, it holds that $Y \perp E \mid h(\mathbf{X})$.*

Proof. The decomposition rule for the assumption (i) $R \perp (h(\mathbf{X}), E)$ implies (ii) $R \perp h(\mathbf{X})$. To simplify notation, we define $Z := h(\mathbf{X})$. Because T is invertible in Y and due to the change of variables (c.o.v.) formula, we obtain

$$p_{Y|Z,E}(y \mid z, e) \overset{(c.o.v.)}{=} p_{R|Z,E}(T(y,z) \mid z, e) \left|\det \frac{\partial T}{\partial y}(y,z)\right|$$
$$\overset{(i)}{=} p_R(r) \left|\det \frac{\partial T}{\partial y}(y,z)\right| \overset{(ii)}{=} p_{R|Z}(r \mid z) \left|\det \frac{\partial T}{\partial y}(y,z)\right| \overset{(c.o.v.)}{=} p_{Y|Z}(y \mid z).$$

□

This implies $Y \perp E \mid Z$. The theorem states in particular that if there exists a suitable diffeomorphism T such that $R \perp (h(\mathbf{X}), E)$, then $h(\mathbf{X})$ satisfies the invariance property with respect to Y. We use Theorem 1 in order to *learn* features h that meet this requirement. In the following, we denote a conditional normalizing flow parameterized via $\boldsymbol{\theta}$ with $T_{\boldsymbol{\theta}}$. Furthermore, $h_{\boldsymbol{\phi}}$ denotes a feature extractor implemented as a neural network parameterized via $\boldsymbol{\phi}$. We can relax condition $R \perp (h_{\boldsymbol{\phi}}(\mathbf{X}), E)$ by means of the Hilbert Schmidt Independence Criterion (HSIC), a kernel-based independence measure (see Subsection 5.3.4). This loss, denoted as $\mathcal{L}_I$, penalizes dependence between the distributions of R and $(h_{\boldsymbol{\phi}}(\mathbf{X}), E)$. The HSIC guarantees that

$$\mathcal{L}_I\Big(P_R, P_{h_{\boldsymbol{\phi}}(\mathbf{X}),E}\Big) = 0 \iff R \perp (h_{\boldsymbol{\phi}}(\mathbf{X}), E) \tag{5.19}$$

where $R = T_{\boldsymbol{\theta}}(Y; h_{\boldsymbol{\phi}}(\mathbf{X}))$ and $P_R, P_{h_{\boldsymbol{\phi}}(\mathbf{X}),E}$ are the distributions implied by the parameter choices $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$. Due to Theorem Theorem 1, minimization of $\mathcal{L}_I(P_R, P_{h_{\boldsymbol{\phi}}(\mathbf{X}),E})$ with respect to $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ will thus approximate the desired invariance property $Y \perp E \mid h_{\boldsymbol{\phi}}(\mathbf{X})$, with exact validity upon perfect convergence. When $R \perp (h_{\boldsymbol{\phi}}(\mathbf{X}), E)$ is fulfilled, the decomposition rule implies $R \perp E$ as well. However, if the differences between environments are small, empirical convergence is accelerated by adding a Wasserstein loss which enforces the latter (see Appendix A.2.2 and Subsection 5.5.2).

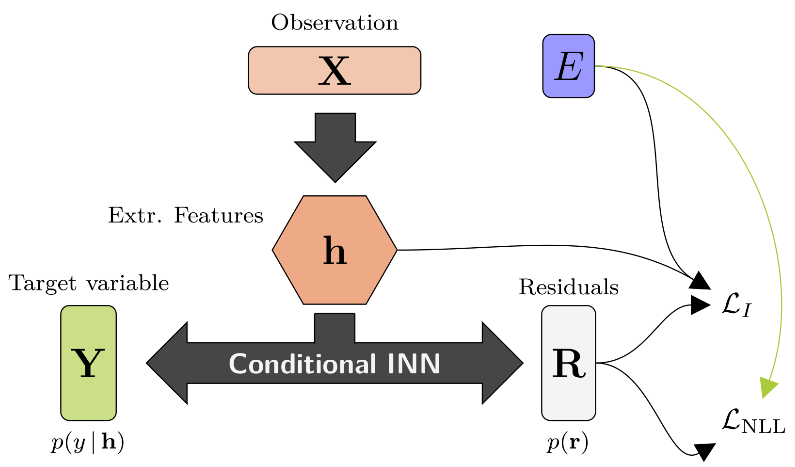


Figure 5.2. Illustration of Conditional Invertible Neural Network (Conditional INN) which optimizes Equation 5.20. h is a feature extractor implemented as a feed-forward neural network. $\mathcal{L}_I$ is the invariance loss that measures the dependence between residuals R and $(E, h(\mathbf{X}))$ and $\mathcal{L}_{\text{NLL}}$ is the negative log-likelihood as in Equation 5.5.

5.4.2. Exploiting Invariances for Prediction

Equation 5.4 can be re-formulated as a differentiable loss using a Lagrange multiplier λ_I on the HSIC loss. λ_I acts as a hyperparameter to adjust the trade-off between the invariance property of $h_{\boldsymbol{\phi}}(\mathbf{X})$ with respect to Y and the mutual information between $h_{\boldsymbol{\phi}}(\mathbf{X})$ and Y. In the following, we consider normalizing flows in order to optimize Equation 5.4. Using Lemma 1 (a), we maximize $\min_{e \in \mathcal{E}} I(Y^e; h_{\boldsymbol{\phi}}(\mathbf{X}^e))$ by minimizing $\max_{e \in \mathcal{E}} \{\mathcal{L}_{\text{NLL}}(T_{\boldsymbol{\theta}}; h_{\boldsymbol{\phi}})\}$ with respect to $\boldsymbol{\phi}, \boldsymbol{\theta}$. To achieve the described trade-off between goodness-of-fit and invariance, we therefore optimize

$$\arg\min_{\boldsymbol{\theta}, \boldsymbol{\phi}} \left(\max_{e \in \mathcal{E}} \left\{ \mathcal{L}^e_{\text{NLL}}(T_{\boldsymbol{\theta}}, h_{\boldsymbol{\phi}}) \right\} + \lambda_I \mathcal{L}_I(P_R, P_{h_{\boldsymbol{\phi}}(\mathbf{X}), E}) \right) \tag{5.20}$$

where $R^e = T_{\boldsymbol{\theta}}(Y^e, h_{\boldsymbol{\phi}}(\mathbf{X}^e))$ and $\lambda_I > 0$. The first term maximizes the mutual information between $h_{\boldsymbol{\phi}}(\mathbf{X})$ and Y in the environment where the features are least informative about Y and the second term aims to ensure an invariant relation. Figure 5.2 illustrates the network that optimizes Equation 5.20 and Algorithm 1 the algorithmic details of the training process.

In the special case that the data is governed by additive noise, Equation 5.20 simplifies under certain assumptions to

$$\arg\min_{\boldsymbol{\theta}} \left(\max_{e \in \mathcal{E}_{\text{seen}}} \left\{ \mathbb{E}\left[(Y^e - f_{\boldsymbol{\theta}}(\mathbf{X}^e))^2 \right] \right\} + \lambda_I \mathcal{L}_I(P_R, P_{f_{\boldsymbol{\theta}}(\mathbf{X}), E}) \right) \tag{5.21}$$

where $R^e = Y^e - f_{\boldsymbol{\theta}}(\mathbf{X}^e)$ and $\lambda_I > 0$. Here, $\arg\max_{\boldsymbol{\theta}} I(f_{\boldsymbol{\theta}}(\mathbf{X}^e), Y^e)$ corresponds to the argmin of the L2-Loss in the corresponding environment. In Algorithm 2 we show how the final model is then optimized. The connection between additive noise models and normalizing flows is derived in the following remark:

Algorithm 1: Training procedure to optimize Equation 5.20. This constitutes the most general case with normalizing flows.

```
Data: Samples from $P_{\mathbf{X}^e,Y^e}$ across seen environments $e \in \mathcal{E}_{\text{seen}}$;
Input: Model parameters $\boldsymbol{\theta}$, $\boldsymbol{\phi}$, number of iterations $n$, and environment-specific
       mini-batch size $m$;
1 for $k = 1, \dots, n$ do
2     for $e \in \mathcal{E}_{seen}$ do
3         Sample mini-batch $\mathcal{B}^e = \{(y_1^e, \mathbf{x}_1^e), \dots, (y_m^e, \mathbf{x}_m^e)\}$ from $P_{Y,\mathbf{X}|E=e}$ for
          $e \in \mathcal{E}_{\text{seen}}$;
4         Compute $r_j^e = T_{\boldsymbol{\theta}}(y_j^e; h_{\boldsymbol{\phi}}(\mathbf{x}_j^e))$ for all $j = 1, \dots, m$;
5     end
6     Update $\boldsymbol{\theta}$, $\boldsymbol{\phi}$ by descending alongside the stochastic gradient
          $\nabla_{\theta,\phi}\Big( \max_{e\in\mathcal{E}_{\text{seen}}} \Big\{ \sum_{i=1}^{m} \Big[\frac{1}{2}\|T_{\boldsymbol{\theta}}(y_i^e; h_{\boldsymbol{\phi}}(\mathbf{x}_i^e))\|^2 - \log \nabla_y T_{\boldsymbol{\theta}}(y_i^e; h_{\boldsymbol{\phi}}(\mathbf{x}_i^e))\Big]\Big\}$
          $+ \lambda_I \mathcal{L}_I(\{r_j^e\}_{j,e}, \{h_{\boldsymbol{\phi}}(\mathbf{x}_j^e), e\}_{j,e})\Big)$;
7 end
Output: In case of convergence, we obtain an invariant feature $h_{\boldsymbol{\phi}^\star}(\mathbf{X})$ and a
        normalizing flow $T_{\boldsymbol{\theta}^\star}$ that represents $P(Y \mid h_{\boldsymbol{\phi}^\star}(\mathbf{X})$;
```

Remark 11. *Let $f_{\boldsymbol{\theta}}$ be a regression function. Solving for the noise term gives $R = Y - f_{\boldsymbol{\theta}}(\mathbf{X})$ which corresponds to a diffeomorphism in Y, namely $T_{\boldsymbol{\theta}}(Y; X) = Y - f_{\boldsymbol{\theta}}(\mathbf{X})$. If we make two simplified assumptions: (i) the noise is Gaussian with zero mean and (ii) $R \perp f_{\boldsymbol{\theta}}(\mathbf{X})$, then we obtain*

$$I(Y; f_{\boldsymbol{\theta}}(\mathbf{X})) = H(Y) - H(Y \mid f_{\boldsymbol{\theta}}(\mathbf{X})) = H(Y) - H(R \mid f_{\boldsymbol{\theta}}(\mathbf{X})) \tag{5.22}$$

$$\overset{(ii)}{=} H(Y) - H(R) \overset{(i)}{=} H(Y) - 1/2 \log(2\pi e \sigma^2) \tag{5.23}$$

where $\sigma^2 = \mathbb{E}[(Y - f_{\boldsymbol{\theta}}(\mathbf{X}))^2]$. In this case maximizing the mutual information $I(Y; f_{\boldsymbol{\theta}}(\mathbf{X}))$ amounts to minimizing $\mathbb{E}[(Y - f_{\boldsymbol{\theta}}(\mathbf{X}))^2]$ with respect to $\boldsymbol{\theta}$, i.e. the standard L2-loss for regression problems. From these considerations, we obtain an approximation of Equation 5.20 via Equation 5.21.

Alternatively we can view the problem as to find features $h_{\boldsymbol{\phi}}\colon \mathbb{R}^D \to \mathbb{R}^m$ such that $I(h_{\boldsymbol{\phi}}(\mathbf{X}), Y)$ gets maximized under the assumption that there exists a model $f_{\boldsymbol{\theta}}(h_{\boldsymbol{\phi}}(\mathbf{X})) + R = Y$ where R is independent of $h_{\boldsymbol{\phi}}(\mathbf{X})$ and is Gaussian. Given this scenario, we obtain the learning objective

$$\arg\min_{\boldsymbol{\theta},\boldsymbol{\phi}} \left(\max_{e\in\mathcal{E}_{\text{seen}}} \left\{ \mathbb{E}\left[(Y^e - f_{\boldsymbol{\theta}}(h_{\boldsymbol{\phi}}(\mathbf{X}^e)))^2 \right] \right\} + \lambda_I \mathcal{L}_I(P_R, P_{h_{\boldsymbol{\phi}}(\mathbf{X}),E}) \right) \tag{5.24}$$

In this case, we can derive the algorithmic details with minor adaptations to Algorithm 2.

Algorithm 2: Training procedure to optimize Equation 5.21. This constitutes the more specific additive noise case.

```
Data: Samples from $P_{\mathbf{X}^e,Y^e}$ across seen environments $e \in \mathcal{E}_{\text{seen}}$;
Input: Model parameters $\boldsymbol{\theta}$, number of iterations $n$, and environment-specific
       mini-batch size $m$;
1 for $k = 1, \dots, n$ do
2     for $e \in \mathcal{E}_{seen}$ do
3         Sample minibatch $\mathcal{B}^e = \{(y_1^e, \mathbf{x}_1^e), \dots, (y_m^e, \mathbf{x}_m^e)\}$ from $P_{Y,\mathbf{X}|E=e}$ for
          $e \in \mathcal{E}_{\text{seen}}$;
4         Compute $r_j^e = y_j^e - f_{\boldsymbol{\theta}}(\mathbf{x}_j^e)$;
5     end
6     Update $\boldsymbol{\theta}$ by descending alongside the stochastic gradient
          $\nabla_{\boldsymbol{\theta}} \left( \max_{e \in \mathcal{E}_{\text{seen}}} \left\{ \sum_{i=1}^{m} |r_j^e|^2 \right\} + \lambda_I \mathcal{L}_I(\{r_j^e\}_{j,e}, \{f_{\boldsymbol{\theta}}(\mathbf{x}_j^e), e\}_{j,e}) \right)$;
7 end
Output: In case of convergence, we obtain a model $f_{\boldsymbol{\theta}^\star}$ minimizing
        Equation 5.21;
```

For the classification case, we consider the expected cross-entropy loss

$$-\mathbb{E}_{\mathbf{X},Y}\left[f(\mathbf{X})_Y - \log\left(\sum_c \exp\left(f(\mathbf{X})_c \right) \right) \right] \tag{5.25}$$

where $f : \mathcal{X} \to \mathbb{R}^m$ returns the logits. Minimizing the expected cross-entropy loss amounts to maximizing the mutual information between $f(\mathbf{X})$ and Y (see Section 4.3 or [199, 200, eq. 3]). We set $T(Y; f(\mathbf{X})) = Y \cdot \text{softmax}(f(\mathbf{X}))$ with component-wise multiplication. Then T is invertible in Y conditioned on the softmax output and therefore Theorem 1 is applicable. Now we can apply the same invariance loss as above in order to obtain a solution to Equation 5.4. In the classification case, we can directly obtain the algorithm's specifics with minor modifications to Algorithm 1.

5.4.3. Relation to Causal Discovery

Under certain conditions, solving Equation 5.4 leads to features which correspond to the direct causes of Y (identifiability). In this case we obtain the causal mechanism by computing the conditional distribution of Y given the direct causes. Hence Equation 5.4 can be seen as an approximation of the causal mechanism when the identifiability conditions are met. The following Proposition states the conditions when the direct causes of Y can be found by exploiting Theorem 1.

Proposition 7. *We assume that the underlying causal graph G is faithful with respect to $P_{\tilde{\mathbf{X}},E}$. We further assume that every child of Y in G is also a child of E in G. A variable selection $h(\mathbf{X}) = \mathbf{X}_S$ corresponds to the direct causes if the following conditions are met:*

(i) $T(Y; h(\mathbf{X})) \perp E, h(\mathbf{X})$ *is satisfied for a diffeomorphism* $T(\cdot; h(\mathbf{X}))$,

(ii) $h(\mathbf{X})$ *is maximally informative about* Y, *and*

(iii) $h(\mathbf{X})$ *contains only variables from the Markov blanket of* Y.

The Markov blanket of Y is the only set of vertices that are necessary to predict Y (see Definition 10 and Proposition 5). In the following, we give a proof of Proposition 7.

Proof. Let $S(\mathcal{E}_{\text{seen}})$ denote a subset of $\mathbf{X}$ which corresponds to the variable selection due to h. Without loss of generality, we assume $S(\mathcal{E}_{\text{seen}}) \subset \mathbf{M}$ where $\mathbf{M}$ is the Markov Blanket. This assumption is reasonable since we have $Y \perp \mathbf{X} \setminus \mathbf{M} \mid \mathbf{M}$ in the asymptotic limit.

Since $pa(Y)$ cannot contain colliders between Y and E, we obtain that $Y \perp E \mid S(\mathcal{E}_{\text{seen}})$ implies $Y \perp E \mid (S(\mathcal{E}_{\text{seen}}) \cup pa(Y))$. This means using $pa(Y)$ as predictors does not harm the constraint in the optimization problem. Due to faithfulness and since the parents of Y are directly connected to Y, we obtain that $pa(Y) \subset S(\mathcal{E}_{\text{seen}})$.

For each subset $\mathbf{X}_S \subset \mathbf{X}$ for which there exists an $X_i \in \mathbf{X}_S \cap \mathbf{X}_{ch(Y)}$, we have $\mathbf{X}_S \not\perp Y \mid E$. This follows from the fact that X_i is a collider, in particular $E \rightarrow X_i \leftarrow Y$. Conditioning on X_i leads to the result that Y and E are not d-separated anymore. Hence, we obtain $Y \not\perp \mathbf{X}_S \mid E$ due to the faithfulness assumption. Consequentially, for each $\mathbf{X}_S$ with $Y \perp E \mid \mathbf{X}_S$ we have $\mathbf{X}_S \cap \mathbf{X}_{ch(Y)} = \emptyset$ and therefore $\mathbf{X}_{ch(Y)} \cap S(\mathcal{E}_{\text{seen}}) = \emptyset$.

Since $\mathbf{X}_{pa(Y)} \subset S(\mathcal{E}_{\text{seen}})$, we obtain that $Y \perp \mathbf{X}_{pa(ch(Y))} \mid \mathbf{X}_{pa(Y)}$ and therefore the parents of $ch(Y)$ are not in $S(\mathcal{E}_{\text{seen}})$ except when they are parents of Y.

Therefore, we obtain that $S(\mathcal{E}_{\text{seen}}) = \mathbf{X}_{pa(Y)}$.

□

One might argue that the condition that every child of Y is also a child of E, i.e. $ch(Y) \subset ch(E)$, is very strict in order to obtain the true direct causes. But this condition is necessary if we do not impose additional constraints on the true underlying causal mechanisms, e.g., linearity [29]. For instance, if $E \rightarrow X_1 \rightarrow Y \rightarrow X_2$, our model would opt for selecting X_1 and X_2 as variables. This follows from the causal Markov condition which implies $E \perp Y \mid X_1, X_2$. Nevertheless, including X_2 into the set of selected variables might be beneficial as we elaborated in Remark 8.

To facilitate explainability and explicit causal discovery, we employ the same gating function and complexity loss as in [201] (see also Appendix A.2.1). The architecture is depicted in Figure 5.3 for three input variables. The gating function h_Φ is a 0-1 mask that marks the selected variables, and the complexity loss $\mathcal{L}(h_\Phi)$ is a soft counter of the selected variables. Intuitively speaking, if we search for a variable selection that conforms to the conditions in Proposition 7, the complexity loss will exclude all non-task relevant variables. Therefore, if $\mathcal{H}$ is the set of gating functions, then $h^\star$ in Equation 5.4 corresponds to the direct causes of Y under the conditions listed in Proposition 7. The complexity loss as well as the gating function can be optimized by gradient descent.

5.5. Experiments

The main focus of this work is on the theoretical and methodological improvements of causality-based domain generalization using information theoretical concepts. A complete and rigorous quantitative evaluation is beyond the scope of this work. In the following we demonstrate proof-of-concept experiments.

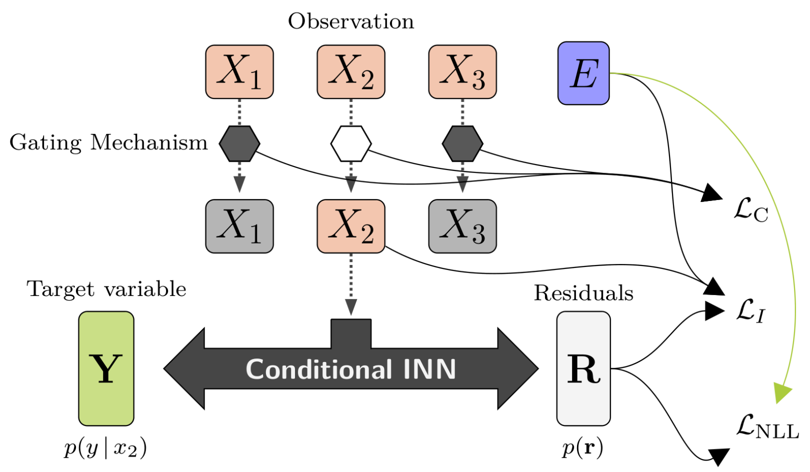


Figure 5.3. Illustration of Conditional Invertible Neural Network (Conditional INN) with gating mechanism. This architecture optimizes Equation 5.20 where the feature space is restricted to gating mechanisms. $\mathcal{L}_I$ is the invariance loss that measures the dependence between residuals R and $(E, h(\mathbf{X}))$, $\mathcal{L}_{\text{NLL}}$ is the negative log-likelihood as in Equation 5.5 and $\mathcal{L}_C$ a complexity loss that enforces to select only relevant variables.

5.5.1. Synthetic Causal Graphs

To evaluate our methods for the regression case, we follow the experimental design of [195]. It rests on the causal graph in Figure 5.5. Each variable $X_1, ..., X_6$ is chosen as the regression target Y in turn, so that a rich variety of local configurations around Y is tested. The corresponding structural equations are selected among four model types of the form $f(\mathbf{X}_{pa(i)}, N_i) = \sum_{j\in pa(i)} \texttt{mech}(a_j X_j) + N_i$, where `mech` is either the identity (hence we get a linear Structural Causal Model (SCM)), Tanhshrink, Softplus or ReLU, and one multiplicative noise mechanism of the form $f_i(\mathbf{X}_{pa(i)}, N_i) = (\sum_{j\in pa(i)} a_j X_j) \cdot (1 + (1/4)N_i) + N_i$, resulting in 1365 different settings. For each setting, we define one observational environment (using exactly the selected mechanisms) and three interventional ones, where soft or do-interventions are applied to non-target variables according to Assumptions 1 and 2 (full details in Section A.3). Each inference model is trained on 1024 realizations of three environments, whereas the fourth one is held back for DG testing. The tasks are to identify the parents of the current target variable Y, and to train a transferable regression model based on this parent hypothesis. We measure performance by the accuracy of the detected parent sets and by the L2 regression errors relative to the regression function using the ground-truth parents. We evaluate four models derived from our theory: two normalizing flows as in Equation 5.20 with and without gating mechanisms (FlowG, Flow) and two additive noise models, again with and without gating mechanism (ANMG, ANM), using a feed-forward network with the objective in Equation 5.24 (ANMG) and Equation 5.21 (ANM).

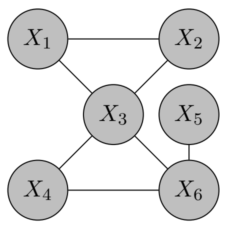


Figure 5.5. Directed graph of our SCM. Target variable Y is chosen among $X_1, \ldots, X_6$ in turn.

For comparison, we train three baselines: ICP (a causal discovery algorithm also exploiting ICM, but restricted to linear regression. See [29] or Section 3.9), a variant of the

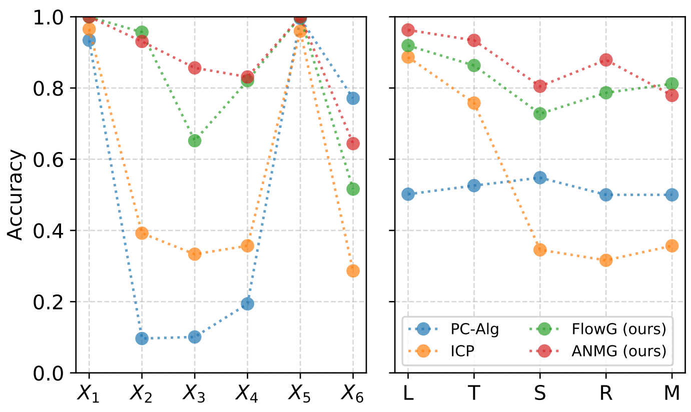


Figure 5.4. Detection accuracy of the direct causes for baselines and our gating architectures, broken down for different target variables (left) and mechanisms (right: **L**inear, **T**anhshrink, **S**oftplus, **R**eLU, **M**ultipl. Noise)

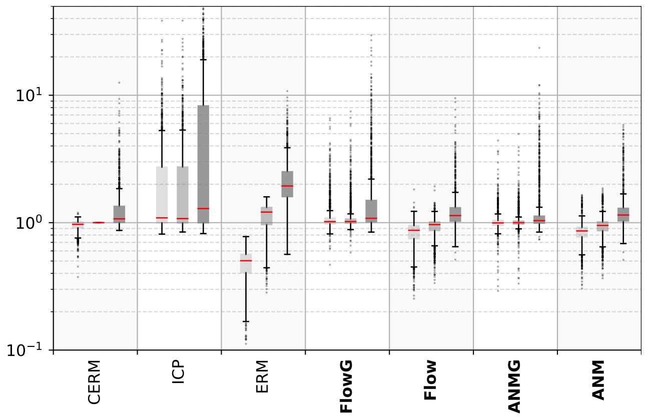


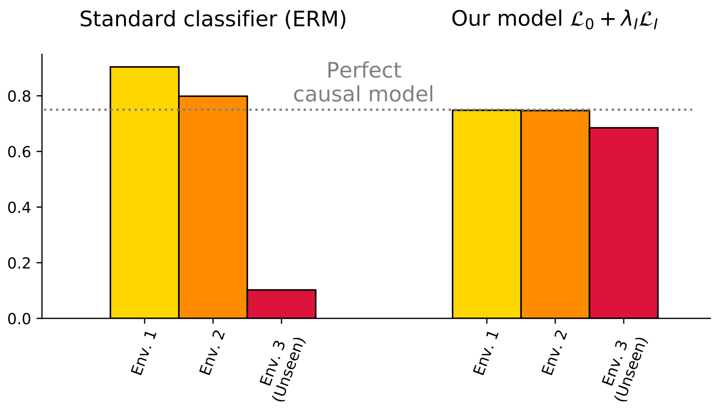
Figure 5.6. Logarithmic plot of L2 errors, normalized by CERM test error. For each method (ours in bold) from left to right: training error, test error on seen environments, domain generalization error on unseen environments.

PC-Algorithm (PC-Alg, see Section 3.9 and Appendix A.3.4) and standard empirical-risk-minimization ERM, a feed-forward network minimizing the L2-loss, which ignores the causal structure by regressing Y on all other variables. We normalize our results with a ground truth model (CERM), which is identical to ERM, but restricted to the true causal parents of the respective Y. The accuracy of parent detection is shown in Figure 5.4 The score indicates the fraction of the experiments where the exact set of all causal parents was found and all non-parents were excluded. We see that the PC algorithm performs unsatisfactorily, whereas ICP exhibits the expected behavior: it works well for variables without parents and for linear SCMs, i.e. exactly within its specification. Among our models, only the gating ones explicitly identify the parents. They clearly outperform the baselines, with a slight edge for ANMG, as long as its assumption of additive noise is fulfilled. Figure 5.6 and Table 5.1 report regression errors for seen and unseen environments, with CERM indicating the theoretical lower bound. The PC algorithm is excluded from this experiment due to its poor detection of the direct causes. ICP wins for linear SCMs, but otherwise has largest errors, since it cannot accurately account for non-linear mechanisms. ERM gives reasonable test errors (while overfitting the training data), but generalizes poorly to un-

Table 5.1. Medians and upper 95% quantiles for domain generalization L2 errors (i.e. on unseen environments) for different model types and data-generating mechanisms (lower is better).

Models	Linear	Tanhshrink	Softplus	ReLU	Mult. Noise
FlowG (ours)	1.05...4.2	1.08...4.8	1.09...5.52	1.08...5.7	1.55...8.64
ANMG (ours)	1.02...1.56	**1.03**...2.23	**1.04**...4.66	**1.03**...4.32	1.46...4.22
Flow (ours)	1.08...1.61	1.14...1.57	1.14...1.55	1.14...1.54	**1.35**...4.07
ANM (ours)	1.05...1.52	1.15...1.47	1.14...1.47	1.15...1.54	1.48...4.19
ICP (Peters et al., 2016)	**0.99**...25.7	1.44...20.39	3.9...23.77	4.37...23.49	8.94...33.49
ERM	1.79...3.84	1.89...3.89	1.99...3.71	2.01...3.62	2.08...5.86
CERM (true parents)	1.06...1.89	1.06...1.84	1.06...2.11	1.07...2.15	1.37...5.1

seen environments, as expected. Our models perform quite similarly to CERM. We again find a slight edge for ANMG, except under multiplicative noise, where ANMG's additive noise assumption is violated and Flow is superior. All methods (including CERM) occasionally fail in the domain generalization task, indicating that some DG problems are more difficult than others, e.g. when the differences between seen environments are too small to reliably identify the invariant mechanism or the unseen environment requires extrapolation beyond the training data boundaries. Models without gating (Flow, ANM) seem to be slightly more robust in this respect. A detailed analysis of our experiments can be found in Section A.3.



	Env. 1	Env. 2	Env. 3
ERM	90.3	79.9	10.2
$\mathcal{L}_0 + \lambda_I \mathcal{L}_I$	74.8	74.7	68.5

Figure 5.7. Accuracy of a standard classifier (ERM) compared to our model on all three environments on the ColoredMNIST data set.

5.5.2. Colored MNIST

To demonstrate that our model is able to perform DG in the classification case, we use the same data generating process as in the colored variant of the MNIST-dataset established by [34], but create training instances online rather than upfront. The response is reduced to two labels – 0 for all images with digit $\{0, \ldots, 4\}$ and 1 for digits $\{5, \ldots 9\}$ – with deliberate label noise that limits the achievable shape-based classification accuracy to 75%. To confuse the classifier, digits are additionally colored such that colors are spuriously associated with the true labels at accuracies of 90% resp. 80% in the first two environments, whereas the association is only 10% correct in the third environment. A classifier naively trained on the first two environments will identify color as the best predictor, but will perform terribly when tested on the third environment. In contrast, a robust model will ignore the unstable relation between colors and labels and use the invariant relation, namely the one between digit shapes and labels, for prediction. We supplement the HSIC loss with a Wasserstein term to explicitly enforce $R \perp E$, i.e. $\mathcal{L}_I = \text{HSIC} + \text{L2}(\text{sort}(R^{e_1}), \text{sort}(R^{e_2}))$ (see Appendix A.2.2). This gives a better training signal as the HSIC alone, since the difference in label-color association between environments 1 and 2 (90% vs. 80%) is deliberately chosen very small to make the task hard to learn. Experimental details can be found in Section A.4. Figure 5.7 shows the results for our model: Naive training ($\lambda_I = 0$, i.e. invariance of residuals is not enforced) gives accuracies corresponding to the association between colors and labels and thus completely fails in test environment 3. In contrast, our model performs close to the best possible rate for invariant classifiers in environments 1 and 2 and still achieves 68.5% in environment 3. This is essentially on par with preexisting methods. For instance, IRM achieves 71% on the third environment for this particular dataset, although the dataset itself is not particularly suitable for meaningful quantitative comparisons. Figure 5.8 demonstrates the trade-off between goodness of fit in the training environments 1 and 2 and the robustness of the resulting classifier: the model's ability to perform DG to the unseen environment 3 improves as λ_I increases. If λ_I is too large, it dominates the classification training signal and performance breaks down in all environments. However, the choice of λ_I is not critical, as good results are obtained over a wide range of settings.

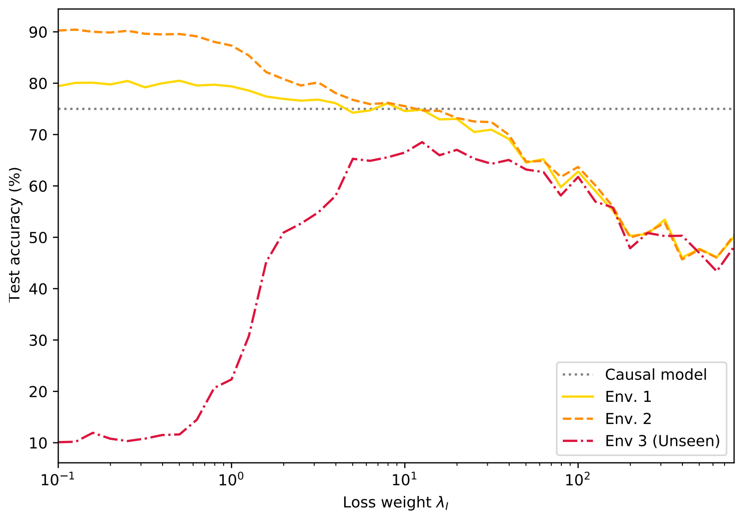


Figure 5.8. Performance of the model in the three environments of the ColoredMNIST data set, depending on the hyperparameter λ_I.

5.6. Discussion

In this work, we have introduced a new method to find invariant and causal models by exploiting the principle of ICM. Our method works by gradient descent in contrast to combinatorial optimization procedures. This circumvents scalability issues and allows us to extract invariant features even when the raw data representation is not in itself meaningful (e.g. we only observe pixel values). In comparison to alternative approaches, our use of normalizing flows places fewer restrictions on the underlying true generative process. We have also shown under which circumstances our method guarantees to find the underlying causal model. Moreover, we demonstrated theoretically and empirically that our method is able to learn robust models with respect to distribution shifts. Future work might include ablations studies in order to improve the understanding of the influence of single components, e.g. the choice of the maxmin objective over the average mutual information or the Wasserstein loss and the HSIC loss. Another interesting direction is to examine our approach in more complex scenarios where, for instance, the invariance assumption may only hold approximately.

6 Context-Aware Domain Generalization and ProDAS

In this chapter, we directly adapt our works from [40] and [39]. In [40], one of our experiments relies on a dataset generated using ProDAS, which we present first.

6.1. ProDAS – Probabilistic Dataset of Abstract Shapes

Abstract We introduce a novel and comprehensive dataset, named ProDAS, which enables the generation of diverse objects with varying shape, size, rotation, and texture/color through a latent factor model. ProDAS offers complete access and control over the data generation process, serving as an ideal environment for investigating disentanglement, causal discovery, out-of-distribution detection, and numerous other research questions. We provide pre-defined functions for the important cases of creating distinct and interconnected distributions, allowing the investigation of distribution shifts and other intriguing applications. The library can be found at `https://github.com/XarwinM/ProDAS`.

6.2. Brief Introduction of ProDAS

Probabilistic Dataset of Abstract Shapes (ProDAS) is a versatile library that provides a customizable latent factor model applicable to any rendering function. This is schematically illustrated in Figure 6.1. The library consists of two parts: Firstly, a customizable latent factor model. For instance, the library enables defining a distribution over object types (e.g. squares or triangles), their color and texture as well as the background. Samples drawn from this distribution can be processed through a renderer to generate the final images. As this distribution is predefined, we can evaluate the likelihood of the rendered images. The library offers also support managing multiple different distributions at the same time, for instance in-distribution and out-of-distributions, or different environments as in Figure 6.2 or Figure 6.3.

As the second component of the library, ProDAS provides "Dsprites++" as a default rendering frontend, supporting colors, textures, and more. By default ProDAS offers different shapes similar to Dsprites [202], supporting also colors and textures. The sensible default that ProDAS provides encompasses the following variables:

- object shape $o_{\text{shape}} \in \{\text{circle}, \text{square}, \text{triangle}\}$
- object size $o_{\text{size}} \in \mathbb{R}_{>0}$
- object position $o_{\text{position}} \in [a, b]^2$

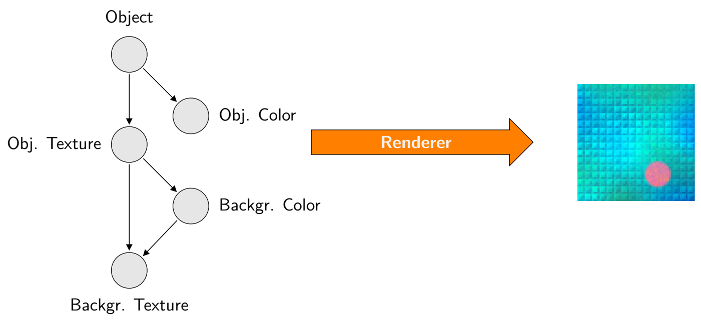


Figure 6.1. An illustration of ProDAS. A distribution over a latent variable model is defined. From this distribution, an instance is sampled, comprising attributes such as object color and texture. This instance is then processed through the renderer, resulting in data accessible to models. Consequently, ProDAS facilitates the sampling of high-dimensional complex data for which the ground truth is known.

- object rotation $o_{\text{rotation}} \in [0, 360]$
- object and background color (e.g., in RGB)
- 9 different foreground and background textures

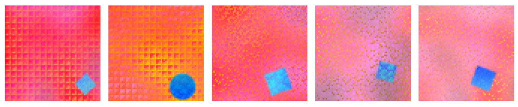
6.3. ProDAS – Target Applications

ProDAS offers the capability to alter both the latent model and rendering functions, enabling the creation of numerous intriguing applications and scenarios. In the following section, we introduce four specific scenarios with the default rendering function. For more challenging applications,the factor model can be applied to a different rendering function such as VirtualKitti [203], Carla [204], etc.

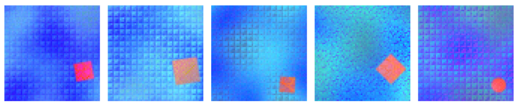
Causal Discovery in Latent Space Given complete access to the latent factors that generate the data, we can presume a latent causal model, thereby delving into the task of causal discovery. In the realm of causal discovery, there is often the assumption that the relevant causal variables are predetermined. However, in our scenario, we consider the task of identifying the underlying causal graph from variables devoid of intrinsic meaning, such as pixels. For instance, the task might involve uncovering the latent causal model as depicted in Figure 6.1. Additionally, this setting aligns with the broader objective of the *disentanglement* task where the latent factors are often assumed to be jointly independent.

Out-of-distribution, but why? In Figure 6.2 we showcase three different out-of-distribution (OOD) scenarios with respect to the *color*, *position*, or *shape* of the objects. In this case, we have created a scenario where one could evaluate an OOD detection algorithm under different conditions. With these scenarios, we enable OOD detection algorithms to show their capabilities in the field of explainability: The samples in Figure 6.2 are OOD for different reasons. Some samples are OOD due to low-level features such as color and some are OOD due to more high-level features such as position. Therefore we can assess a model's ability to understand differences between OOD-ness.

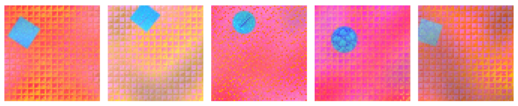
Distribution Shift Furthermore, we can explore scenarios involving distribution shifts. In Figure 6.3, we defined four domains, each sharing identical characteristics except for their varying appearance across domains. This example showcases the potential for exploring distribution shifts, which can manifest in different forms.



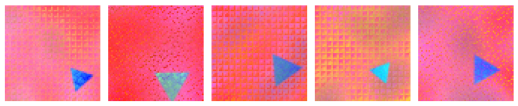
(a) Training Data.



(b) OOD due to Color.

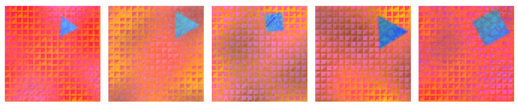


(c) OOD due to Position.

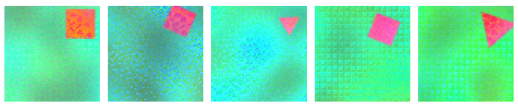


(d) OOD due to shape

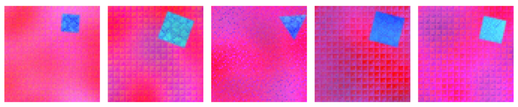
Figure 6.2. ProDAS enables the support for multiple distributions concurrently. In this figure, we implemented one in-distribution scenario and three distinct out-of-distribution (OOD) situations.



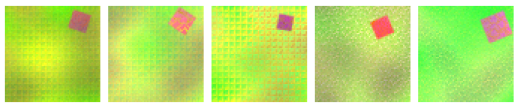
(a) Domain 1



(b) Domain 2



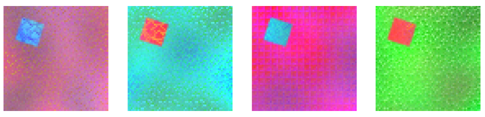
(c) Domain 3



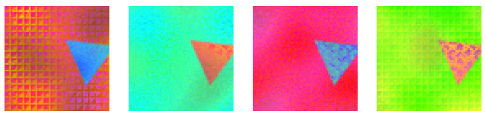
(d) Domain 4

Figure 6.3. ProDAS provides support for multiple distributions simultaneously. In this figure, we implemented various domains to simulate a distribution shift setting.

Multi-View Learning In a multi-view setting, practitioners have access to various representations (also called multiple views) of the same instance. For example, a book translated into multiple languages provides multiple views of the same content. Likewise, within ProDAS, we can establish a multi-view setting. For instance in Figure 6.4, we observe the same objects in multiple views.



(a) Multiple views of one instance



(b) Multiple views of one instance

Figure 6.4. This figure shows two instances in four views.

Many More In addition to the demonstrated application, ProDAS offers a wide range of potential applications. These encompass tasks such as disentanglement, domain transfer, domain adaptation, few-shot learning, density estimation, among many others.

6.4. Towards Context-Aware DG: Representing Environments with Permutation-Invariant Networks

The subsequent sections of this chapter are a direct adaptation from our work in [40].

Abstract In this work, we show that information about the context of an input $\mathbf{X}$ can improve the predictions of deep learning models when applied in new domains or production environments. We formalize the notion of context as a permutation-invariant representation of a set of data points that originate from the same environment/domain as the input itself. These representations are jointly learned with a standard supervised learning objective, providing incremental information about the unknown outcome. Furthermore, we offer a theoretical analysis of the conditions under which our approach can, in principle, yield benefits, and formulate two necessary criteria that can be easily verified in practice. Additionally, we contribute insights into the kind of distribution shifts for which our approach promises robustness. Our empirical evaluation demonstrates the effectiveness of our approach for both low-dimensional and high-dimensional data sets. Finally, we demonstrate that we can reliably detect scenarios where a model is tasked with unwarranted extrapolation in out-of-distribution (OOD) domains, identifying potential failure cases. Consequently, we showcase a method to select between the most predictive and the most robust model, circumventing the well-known trade-off between predictive performance and robustness.

6.5. Introduction: Context-Aware DG

Distribution shifts are the cause of many failure cases in machine learning [12, 14] and the root of various peculiar phenomena in classical statistics, such as Simpsons' paradox [32, 122]. In this work, we employ permutation-invariant neural networks as set-encoders [205, 206] to improve the predictions of standard supervised models under distribution shift. Specifically, we consider training in the realm of Domain Generalization (DG), a setting where data from distinct environments[1] is available for training and testing [15, 17].

For illustration, consider a probabilistic model $P(Y \mid \mathbf{X})$ that classifies diseases Y from magnetic resonance (MR) images $\mathbf{X}$. Since MR images are not fully standardized, the classifier should work slightly differently for images acquired by different hardware brands. It thus makes sense to inform the classifier about the current environment E (here: hardware brand) and extend it into $P(Y \mid \mathbf{X}, E)$. This raises two questions:

(1) Under which circumstances will the classifier $P(Y \mid \mathbf{X}, E)$ be superior to $P(Y \mid \mathbf{X})$?

(2) How should E be represented to maximize the performance gain?

The first question is important because there might exist a function $E = f(\mathbf{X})$ allowing the classifier $P(Y \mid \mathbf{X})$ to deduce E from the data $\mathbf{X}$. For example, E might be inferred from the

[1]We use the terms environment and domain interchangeably.

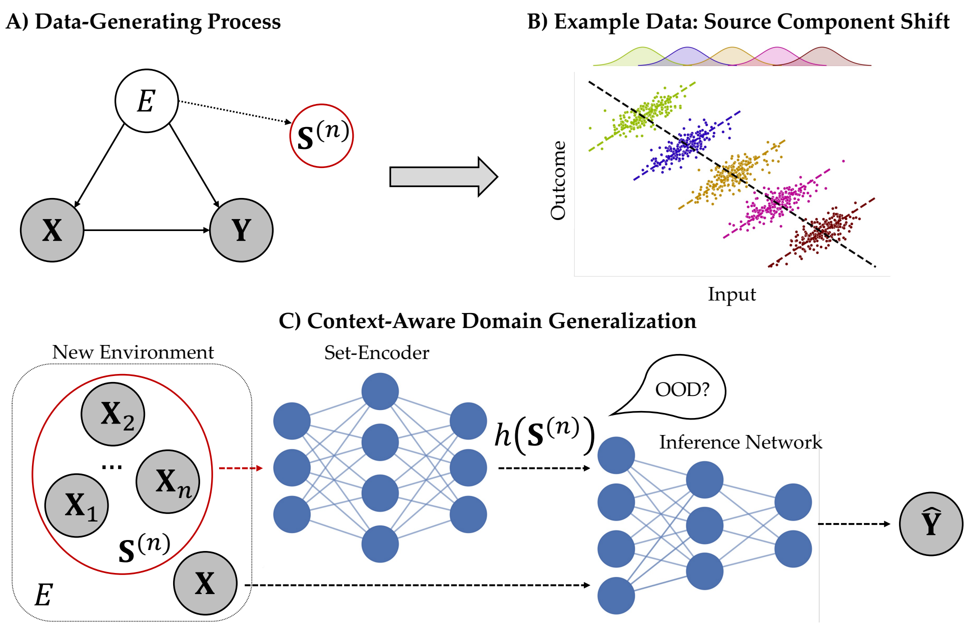


Figure 6.5. Conceptual sketch of our setup and approach. **A)** Data-generating process (DGP) that fulfills our criteria. We assume that the environment E is a source node that is not caused by any system variable and that the relationship between $\mathbf{X}$ and Y varies with the environment. $\mathbf{S}^{(n)}$ is a set of n IID inputs available in the new environment (i.e. context). **B)** The source component shift corresponding to the assumed DGP and example data coinciding with Simpson's paradox. **C)** The workings of our approach in a test environment. A set-encoder generates a permutation-invariant representation $h(\mathbf{S}^{(n)})$ of the context. An inference network (e.g., a classifier) processes the representation along with the target input $\mathbf{X}$ and predicts the unknown outcome (e.g., label) of the target input. The set-representation can be combined with the input to reliably detect out-of-distribution (OOD) queries and prevent failure cases in domain generalization due to model misspecification.

periphery of the given image, while Y depends on its central region. Then, no additional information is gained by passing E explicitly, and both classifiers perform identically.

A straightforward answer to the second question is to distinguish environments by discrete labels. However, we argue that E should be a continuous embedding. First, continuous embeddings can also be computed for new environments that have not been present in the training data. Second, when $P(Y \mid \mathbf{X}, E)$ receives a continuous E, it can potentially configure itself for unseen environments by interpolating between the training environments. And finally, discrete and continuous E are equally informative for the known environments, ensuring no loss in information.

In the current work, we systematically investigate both questions, formalize three criteria when $P(Y \mid \mathbf{X}, E)$ is beneficial, and demonstrate how continuous embeddings E can be learned from auxiliary data by means of set encoders (see Figure 6.5). Notably, two of these criteria are empirically testable using standard models and are shown to be necessary conditions for the success of the approach. First, we require that a single input alone is insufficient to deduce the originating environment (see Criterion 2). If this condition is not met, our model cannot possibly extract more information about the originating environment, compared to a standard model. Second, we require that, if the environment label is known by an *oracle* and given as an additional input to the standard model, its performance must improve (see Criterion 3). Otherwise, the set-based representation of the environment cannot possibly yield benefits. Notably, these two criteria are easy to verify and also met under the *source component shift* (see Subsection 6.6.6), which is a dataset shift that occurs in many scenarios [8].

When test environments are highly dissimilar to the training environments, all DG methods enter an extrapolation regime with unknown prospects of success and the potential to generate silent failures. While our approach is not exempt from this "curse of extrapolation", it comes with a natural way to reliably detect novel environments in set-representation space and delineate its competence region (see our work in Chapter 7). Moreover, we propose a method to select between models that are specialized in the in-distribution (ID) setting vs. models that are robust to out-of-distribution (OOD) scenarios on the fly. Thus, we can overcome the notorious trade-off between ID predictive performance and robustness to distribution shifts [31, 33, 207]. Accordingly, we can adaptively select the most robust model in the OOD setting and the most predictive model in the ID setting, an approach we demonstrate on the ColoredMNIST data set (see Figure 2.1). In summary, our contributions are:

- We propose a novel approach to Domain Generalization (DG) that leverages context information from new environments in the form of learnable set-representations;
- We formalize the necessary and empirically verifiable conditions under which our approach can reap benefits from context information and improve on standard approaches;
- We perform an extensive empirical evaluation and show that we can reliably detect failure cases when the necessary criteria of our theory are not met, or when extrapolation is required.

6.6. Methods

6.6.1. Notation

We denote inputs $\mathbf{x} \in \mathcal{X}$ and outputs as $Y \in \mathcal{Y}$, without any strict requirements on the input and output spaces $\mathcal{X}$ and $\mathcal{Y}$, respectively. We treat the (unknown) domain label E as a random variable and denote with $\mathbf{s}^{(n)}$ an i.i.d. sample (i.e., a set of further inputs) from the given domain. The domain label E is only known during training time and unknown during inference.

6.6.2. General Idea

The key idea of our approach is to build models that utilize not only a singleton input $\mathbf{x}$ to predict a target Y, but also information about the environment of $\mathbf{x}$ that can improve the prediction $\hat{Y}$. Providing environmental information in the form of a one-hot label is hardly feasible, as it presupposes the exact number of possible environments to be known during training, and that we always know from which environment an input originates at inference time. Consequently, such an approach is doomed to fail when the test data originates from a novel or unknown environment.

To overcome this problem, we employ permutation-invariant neural networks to adaptively represent environmental information given a set of test inputs. We will first detail our approach and then discuss criteria under which we can expect to reap benefits from the additional set-representation. Afterwards, we explain the theoretical data-generating process that matches these criteria, providing insight into the distribution shifts for which our approach may prove advantageous in practice. Finally, we discuss the process of identifying new environments that demand extrapolation, potentially leading to failure cases.

6.6.3. Permutation-Invariant Neural Networks

As mentioned above, a basic goal of our approach is to synthesize contextual information about a target input $\mathbf{x}$ by compressing a set of n further inputs $\mathbf{S}^{(n)} := \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ from the same environment into a permutation-invariant representation. The notion of permutation invariance is closely related to a core concept in probabilistic modeling and Bayesian inference – *exchangeability* [208]. Accordingly, an exchangeable sequence of random vectors is characterized by a joint distribution which is invariant to any permutation of the elements:

$$P(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = P(\mathbf{x}_{\pi(1)}, \mathbf{x}_{\pi(2)}, \ldots, \mathbf{x}_{\pi(n)}), \tag{6.1}$$

where $\pi : \mathbb{N} \to \mathbb{N}$ denotes an arbitrary permutation of index elements $n \in \mathbb{N}$.

Exchangeable observations may come in various forms, for instance, patients entering a hospital, different measurements obtained with the same device, or sets of visual images, such as faces in a crowd. When it comes to learning *exchangeable symmetries*, permutation-invariant functions can serve as a key building block in neural architectures [206], as they automatically encode a favorable inductive bias towards permutation-invariance by design. A simple and intuitive way to create permutation-invariant functions involves the sum-decomposition

$$h(\mathbf{S}^{(n)}) = \rho\left(\sum_{i=1}^{n} \sigma(\mathbf{x}_i)\right) \tag{6.2}$$

where σ and ρ can be any functions, including deep neural networks [206, 209]. h is permutation-invariant due to the summation operator which ensures that the argument of ρ is agnostic to the order of elements in the set $\mathbf{S}^{(n)}$.

Despite having favorable theoretical properties, plain sum-decompositions can have limited representational capacity in practice [209, 210]. Thus, more expressive permutation-invariant functions can be learned by stacking equivariant transformations with sum-decompositions [206, 209] or by using a self-attention mechanism without positional encodings [211]. In order to re-assess the expressiveness of these methods, we show a comparison of the binary domain classification accuracy based on domain overlap and set size in Section B.7.

6.6.4. Context-Aware Model

Our model consists of two key components (also illustrated in Figure 6.5):

- a permutation-invariant network h_ψ ("set encoder") with parameters ψ that maps a set input $\mathbf{S}^{(n)}$ to a summary vector $h_\psi(\mathbf{S}^{(n)})$, and
- an inference network f_ϕ with parameters ϕ that maps both the input $\mathbf{x}$ and the summary vector $h_\psi(\mathbf{S}^{(n)})$ to a final prediction.

The complete model is denoted as $f_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{S}^{(n)}) = f_{\boldsymbol{\phi}}(\mathbf{X}, h_{\boldsymbol{\psi}}(\mathbf{S}^{(n)}))$ with parameters $\boldsymbol{\theta} = (\boldsymbol{\psi}, \boldsymbol{\phi})$ for short. For a given supervised learning task, we aim to find the minimum to the following optimization problem

$$\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\mathbf{X},Y,E}\left[c(f_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{S}^{(n)}), Y)\right], \tag{6.3}$$

where c is a task-specific loss function (e.g., cross-entropy for classification or mean squared error for regression). The algorithmic details for optimizing Equation 6.3 are detailed in Algorithm 3. For practical reasons, we first apply a feature extractor g and then pass $\{g(\mathbf{X}) \mid \mathbf{X} \in \mathbf{S}^{(n)}\}$ as input to the set encoder and $g(\mathbf{X})$ as input to the inference network, building upon the features extracted by g. Note, that g can be a pre-trained network, in which case we can treat it as a fixed transformation, or its parameters can be optimized jointly with $\boldsymbol{\theta}$.

Algorithm 3: Optimizing Equation 6.3 for context-aware domain generalization.

```
Data: Samples from the joint distribution $P(\mathbf{X}, Y, E)$;
Input: Composite model parameters $\boldsymbol{\theta}$, set size $n$, batch size $m$, loss-function $c$,
       number of iterations $k$, learning rate schedule $\alpha(k)$;
1 for $i = 1, \dots, k$ do
2     Sample mini-batch $\mathcal{B} = \{(\mathbf{x}_1, \boldsymbol{y}_1, \mathrm{env}_1), \dots, (\mathbf{x}_m, \boldsymbol{y}_m, \mathrm{env}_m)\}$ from
        $P(\mathbf{X}, Y, E)$;
3     for $j = 1, \dots, m$ do
4         Sample set $\mathbf{s}_j^{(n)} = \{\mathbf{x}_1, \dots \mathbf{x}_n\}$ from $P(\mathbf{X} \mid E = \mathrm{env}_j)$;
5         Replace $\mathrm{env}_j$ with $\mathbf{s}_j^{(n)}$ in $\mathcal{B}$;
6     end
7     Update $\boldsymbol{\theta}$ using adaptive mini-batch gradient descent (or any variant):
          $\boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}_{k-1} - \alpha(k)\nabla_{\boldsymbol{\theta}}\left(\sum_{j=1}^{m} c\left(f_{\boldsymbol{\theta}}(\mathbf{x}_j, \mathbf{s}_j^{(n)}), \boldsymbol{y}_j\right)\right)$
8 end
Output: Trained context-aware model $f_{\boldsymbol{\theta}}$;
```

6.6.5. Criteria for Improvement

In the following, we establish criteria under which our method can exploit the distribution shifts between environments and yield improved predictions. In total, we propose three criteria that are necessary to achieve incremental improvement. In Theorem 2, we show how these criteria are related to each other. In the formulations below, $I(\mathbf{X}; Y)$ denotes the *mutual information* between random vectors $\mathbf{X}$ and Y and $I(\mathbf{X}; Y \mid Z)$ denotes the conditional mutual information given a third random vector Z (exact definitions of these concepts can be found in Section 4.3). The symbol $\perp$ (resp. $\not\perp$) between two random vectors $\mathbf{X}$ and Y is used to express that the random vectors are independent (resp. dependent) or conditionally independent (resp. dependent) given a third random vector Z.

First, we require that given an input $\mathbf{X}$, a further set of i.i.d. inputs $\mathbf{S}^{(n)}$ from the same environment provides *incremental information* about Y. This is exactly what we need to achieve improved predictive performance, and we can formally define it as our first criterion:

Criterion 1. $\mathbf{S}^{(n)} \not\perp Y \mid \mathbf{X}$ *or equivalently* $I(\mathbf{S}^{(n)};\, Y \mid \mathbf{X}) > 0$.

The second criterion requires that, given a target input $\mathbf{X}$, a set of further i.i.d. inputs $\mathbf{S}^{(n)}$ from the same environment provides *additional information* about the origin environment of $\mathbf{X}$.

Criterion 2. $E \not\perp \mathbf{S}^{(n)} \mid \mathbf{X}$ *or equivalently* $I(E;\, \mathbf{S}^{(n)} \mid \mathbf{X}) > 0$.

In Figure 6.5, an instance $\mathbf{X}$ cannot be assigned with complete certainty to an environment. Consequentially, further data provides additional information about the environment. In general, the more data we consider, the better we can predict the originating environment. Crucially, this criterion is *not satisfied*, if we can recover the origin environment from the singleton input $\mathbf{X}$ alone.

The third criterion requires that the singleton input $\mathbf{X}$ gains information about Y if we also consider the origin environment E of $\mathbf{X}$.

Criterion 3. $Y \not\perp E \mid \mathbf{X}$ *or equivalently* $I(Y;\, E \mid \mathbf{X}) > 0$.

In Figure 6.5, this is evidently the case: If we knew the environment from which the data stems, we could improve our prediction of Y. Furthermore, this criterion can serve as a sanity check in case we have an oracle that can identify the origin environment of the data with perfect accuracy.

In what follows, we show that Criterion 2 and Criterion 3 are necessary conditions for Criterion 1. We furthermore prove that if we can extract the environment label fully from $\mathbf{S}^{(n)}$, then Criterion 2 and Criterion 3 are sufficient conditions for Criterion 1. We even generalize this result for the case where the environment label is not inferable with 100% accuracy.

Theorem 2. *The following statements hold:*

(a) If $E \perp \mathbf{S}^{(n)} \mid \mathbf{X}$, it follows that $Y \perp \mathbf{S}^{(n)} \mid \mathbf{X}$. This is equivalent to the implication that if Criterion 2 is unattainable, then Criterion 1 is also not satisfied.

(b) If $E \perp Y \mid \mathbf{X}$, we achieve $Y \perp \mathbf{S}^{(n)} \mid \mathbf{X}$. This statement corresponds to: Criterion 3 is a necessary condition for Criterion 1.

(c) Assume that there exists a deterministic function g with $g(\mathbf{S}^{(n)}) = E$, then $Y \not\perp E \mid \mathbf{X}$ implies $Y \not\perp \mathbf{S}^{(n)} \mid \mathbf{X}$. This conveys that if we could perfectly infer E from $\mathbf{S}^{(n)}$, then Criterion 3 implies Criterion 1.

(d) Assume that there exists a function g and a noise variable Z that elicits the relation $E = g(\mathbf{S}^{(n)}) + Z$ and satisfies $\mathbf{S}^{(n)} \perp Z \mid \mathbf{X}$ as well as $\mathbf{S}^{(n)} \perp Z \mid \mathbf{X}, Y$. Furthermore, assume that $Y \not\perp E \mid \mathbf{X}$ and $I(Y; E \mid \mathbf{X}) > I(Z; Y \mid \mathbf{X})$. Then, we achieve $Y \not\perp \mathbf{S}^{(n)} \mid \mathbf{X}$, recovering Criterion 1.

Proof. For the upcoming proofs, we extensively employ the chain rule of mutual information:

$$I(Y; Z, \mathbf{X}) = I(Y; Z \mid \mathbf{X}) + I(Y; \mathbf{X}) \tag{6.4}$$

Additionally, we utilize the inequality $I(Y; \mathbf{S}^{(n)} \mid \mathbf{X}) \leq I(Y; E \mid \mathbf{X})$ which follows from the data processing inequality and how $\mathbf{S}^{(n)}$ relates to the other variables (see Figure 6.5).

For (b): We easily achieve

$$\begin{aligned} I(Y; \mathbf{S}^{(n)}, \mathbf{X}) &= I(Y; \mathbf{S}^{(n)} \mid \mathbf{X}) + I(Y; \mathbf{X}) && (6.5)\\ &\leq I(Y; E \mid \mathbf{X}) + I(Y; \mathbf{X}) && (6.6)\\ &= I(Y; \mathbf{X}) && (6.7)\end{aligned}$$

Therefore, we have

$$0 \leq I(Y; \mathbf{S}^{(n)} \mid \mathbf{X}) = I(Y; \mathbf{S}^{(n)}, \mathbf{X}) - I(\mathbf{X}; Y) \leq 0 \tag{6.8}$$

which proves (b).

For (a): We can write

$$\begin{aligned} I(\mathbf{S}^{(n)}; Y, \mathbf{X}) &= I(\mathbf{S}^{(n)}; Y \mid \mathbf{X}) + I(\mathbf{S}^{(n)}; \mathbf{X}) && (6.9)\\ &\leq I(\mathbf{S}^{(n)}; E \mid \mathbf{X}) + I(\mathbf{S}^{(n)}; \mathbf{X}) && (6.10)\\ &= I(\mathbf{S}^{(n)}; \mathbf{X}) && (6.11)\end{aligned}$$

and therefore

$$0 \leq I(Y; \mathbf{S}^{(n)} \mid \mathbf{X}) = I(\mathbf{S}^{(n)}; Y, \mathbf{X}) - I(\mathbf{X}; \mathbf{S}^{(n)}) \leq 0 \tag{6.12}$$

and conclusively $Y \perp \mathbf{S}^{(n)} \mid \mathbf{X}$.

For (c) is easily seen, using the data processing inequality, that $0 < I(Y; E \mid \mathbf{X}) = I(Y; g(\mathbf{S}^{(n)}) \mid \mathbf{X}) \leq I(Y; \mathbf{S}^{(n)} \mid \mathbf{X})$ and therefore (c) holds true.

For (d), we also employ the entropy $H(\mathbf{X})$ as well as the conditional entropy $H(\mathbf{X} \mid Y)$ (for definitions and elementary properties see Section 4.3). We first establish that

$$I(A + B; C) \leq I(A; C) + I(B; C) \tag{6.13}$$

for any RVs A, B, C with $A \perp B$ and $A \perp B \mid C$:

$$\begin{aligned} I(A + B; C) &= H(A + B) - H(A + B \mid C)\\ &\overset{(\star)}{=} \Big(H(A) + H(B) - H(A \mid A + B)\Big) && (6.14)\\ &\quad - \Big(H(A \mid C) + H(B \mid C) - H(A \mid A + B, C)\Big)\\ &= I(A; C) + I(B; C) - H(A \mid A + B) + H(A \mid A + B, C)\\ &\overset{(\star\star)}{\leq} I(A; C) + I(B; C) && (6.15)\end{aligned}$$

($\star$) follows with the chain rule for entropy

$$H(A, A+B) = H(A) + H(A+B \mid A) \tag{6.16}$$
$$= H(A) + H(B \mid A) \overset{A \perp B}{=} H(A) + H(B) \tag{6.17}$$
$$= H(A+B) + H(A \mid A+B) \tag{6.18}$$

which implies $H(A+B) = H(A) + H(B) - H(A \mid A+B)$ and equally when conditioning on C.

($\star\star$) follows since $h(A \mid A+B, C) \leq h(A \mid A+B)$.

Equation 6.14 can be extended to the conditional mutual information if $A \perp B \mid D$ and $A \perp B \mid D, C$:

$$I(A+B; C \mid D) \leq I(A; C \mid D) + I(B; C \mid D) \tag{6.19}$$

Since $\mathbf{S}^{(n)} \perp Z \mid \mathbf{X}$ and $\mathbf{S}^{(n)} \perp Z \mid \mathbf{X}, Y$, we achieve

$$0 < I(Y; E \mid \mathbf{X}) = I(Y; g(\mathbf{S}^{(n)}) + Z \mid \mathbf{X}) \tag{6.20}$$
$$\leq I(Y; g(\mathbf{S}^{(n)}) \mid \mathbf{X}) + I(Y; Z \mid \mathbf{X}) \tag{6.21}$$
$$\leq I(Y; \mathbf{S}^{(n)} \mid \mathbf{X}) + I(Y; Z \mid \mathbf{X}) \tag{6.22}$$

and therefore

$$0 < I(Y; E \mid \mathbf{X}) - I(Y; Z \mid \mathbf{X}) \leq I(Y; \mathbf{S}^{(n)} \mid \mathbf{X}) \tag{6.23}$$

which concludes the proof. □

A brief discussion of the theorem's presumptions can be found in Section B.1. Unfortunately, we cannot deduce that $Y \perp \mathbf{S}^{(n)} \mid \mathbf{X}$ follows from Criterion 2 and Criterion 3 in general. An example where Criterion 2 and Criterion 3 hold, but Criterion 1 is violated, is provided in the following:

Example 7. Criterion 2 and Criterion 3 are not sufficient to imply Criterion 1. This can be seen in an example with three environments $j \in \{1, 2, 3\}$. Assume the first two have completely identical input distributions. We presume that both input distributions adhere to a uniform distribution$\mathcal{U}[a, b]$. Furthermore, we assume that the third input distribution also follows a uniform distribution that is slightly shifted, i.e. $\mathcal{U}[a + \frac{a+b}{2}, b + \frac{a+b}{2}]$. Due to the overlap between the third and the first two environments, a set input provides additional information about E compared to a single sample X, verifying Criterion 2.

When considering the mechanism linking inputs to outputs, we assume that within the range $[a, \frac{a+b}{2}]$, the relationship between input X and output Y differ – for instance, linear relations with distinct values. Additionally, we presume that within the inverval $(\frac{a+b}{2}, b + \frac{a+b}{2}]$, the relationship between input X and output Y remains consistent across environmenst, e.g., is constant. This aligns with Criterion 3: When the environment is known, we can improve the prediction, specifically within the range $[a, \frac{a+b}{2}]$.

However, Criterion 1 is unsatisfiable. While the set input allows us to effectively distinguish environment 3 (i.e. the one with support$\mathcal{U}[a+\frac{a+b}{2}, b+\frac{a+b}{2}]$) from the other ones, it does not allow us to differentiate between environment 1 and environment 2. Since the relationship between X and output Y differs solely within the supports of environment 1 and environment 2 (specifically, in $\mathcal{U}[a, \frac{a+b}{2}]$), the set input cannot provide additional information about the output Y compared to the single input X. Consequentially, it holds $Y \perp \mathbf{S}^{(n)} \mid X$ and Criterion 1 is not met.

6.6.6. Source Component Shift

Using our approach, we can characterize the kind of distribution shift that allows our criteria to be satifsied. *Source component shift* refers to the scenario where the data comes from a number of sources (or environments) each with different characteristics (see Subsection 3.8.1). The source component shift can be described by the graphical model in Figure 6.5, where the environment directly affects both the input $\mathbf{X}$ and the outcome Y. Problems that conform to the graph in Figure 6.5 have two important implications. First, the input distribution changes whenever the environment changes.[2] Second, the relationship between inputs and outcomes varies with the environment (corresponding to Criterion 1). For more details on this kind of distribution shift, we refer the reader to Subsection 3.8.1 or [8, Chapter 1.9]. It is also worth noting that the graph in Figure 6.5 corresponds to *Simpson's paradox* [32, 122], which supplies a proof-of-concept for our approach (see **Experiment 1**).

6.6.7. Detection of Novel Environments

During test time, data could either originate from an environment that corresponds to one of the training environments (but its origins are unknown) or from a previously unseen environment. In the following, we explain how we aim to detect the second case that might result in potential failure cases due to fundamental challenges in extrapolation. Following our work in Chapter 7, we can define a score $s(h_{\psi}(\mathbf{S}^{(n)}))$ on the summary vector $h_{\psi}(\mathbf{S}^{(n)})$ implicit in our model $f_{\boldsymbol{\theta}}(\mathbf{X}, \mathbf{S}^{(n)})$ that aims to predict the target variable Y. As a score function, we consider the distance of $h_{\psi}(\mathbf{S}^{(n)})$ to the k-nearest neighbors in the training data in the feature space of the set-encoder. Accordingly, set-representations that elicit a score surpassing a certain threshold are considered to originate from a novel environment. Similar to our work in Chapter 7, we consider the score distribution and set a threshold to classify a specific percentage, denoted as q, of in-distribution samples as originating from a known environment. To establish this threshold, we consider the q-th percentile of scores obtained from the validation set. We also compare our novel environment detector with the same score function computed solely from singleton features $g(\mathbf{X})$. These results can be quickly previewed in Table 6.3.

6.7. Related Work

6.7.1. Domain Generalization

The aim of domain generalization (DG) is to train models that generalize well under distribution shifts [15, 17]. The DG setting involves access to data from multiple domains during training exploiting the heterogeneity between domains. In this context, the plethora

[2]In our case, we require a stronger assumption, namely, that the domains are not deducible from a single input, as formalized in Criterion 2.

of algorithms aimed at improving robustness has been divided into three categories as explained in Subsection 2.2.3. Our approach explores a different avenue exploiting contextual information about the origin of the data via an adaptive environment embedding.

In contrast to Domain Adaptation (DA) [52], where samples from the test domain are given during training, in DG no knowledge about the test environment is available during training. As a middle-ground between DA and DG, test-time adaptation (TTA) involves the provision of unlabeled samples during test time, enabling further fine-tuning of the model [212]. TTA can be categorized into test-time domain adaptation, test-time batch adaptation (TTBA), and online test-time adaptation (OTTA) [212].

Among these settings, our work aligns most closely with the TTBA scenario concerning the utilization of environment information. In TTBA a pre-trained model is adapted to one or a few inputs [213–215]. Each adaptation depends on the mini-batch at hand. Similarly, we consider a mini-batch as a set input to deliver contextual information. However, we do not adapt or fine-tune our model at test time, but rather directly extract context information via the set-encoder. This allows us to identify whether extrapolation is necessary, enabling model misspecification detection [216]. Moreover, since we do not need to fine-tune the model at test time, our approach is considerably more efficient.

Finally, [217] assume a setting where inputs from multiple domains are available, but it is unknown which sample belongs to which environment – even during training time. The authors infer potential domain labels that are used for downstream invariant learning, for example, via Invariant Risk Minimization (IRM) [34]. This method improves on baseline models and enables the application of DG methods in the absence of domain labels. Interestingly, it could be combined with our approach when no environment labels are provided during training, and we leave this as an avenue for future research.

6.7.2. Learning Permutation-Invariant Representations

Analyzing set-structured data with neural networks has received much theoretical [206, 210, 218] and empirical [209, 211, 219] momentum in recent years. For instance, [219] build on the set transformer architecture [211] and augment the attentive encoder with the capability to learn dynamic templates for attention-based pooling. The resulting "PICASO" blocks include consecutive multi-head attention stacks with skip connections, which update their template(s) depending on previous template(s) and the particular input set (in contrast to global templates). Differently, [220] propose to learn set-specific representations, along with global "prototypes", using an optimal transport (OT) optimization criterion. The authors also show how to use their criterion for generative and few-shot classification tasks In a somewhat similar vain, [221] investigate a gradient-based optimization method for aggregating set-structured data. Importantly, these methods comprise a pool of algorithms which can be used as the backbone architecture for realizing the set-encoder in our approach.

Notably, the methods above impose no probabilistic structure on the set-representations, since the latter are mainly used as deterministic features for downstream tasks. In contrast, probabilistic models attempt to learn a conditional or a marginal distribution over the set-representations. In the Bayesian literature, sets represent finitely exchangeable sequences, embodying the core probabilistic structure of most Bayesian models [208]. In particular, hierarchical or multi-level Bayesian models are used to model the dependencies in nested

Model	**Symbol**	**Description**	**Purpose**
Context-aware model (ours)	$f^{Y\mid\mathbf{X},\mathbf{S}^{(n)}}$	Predicts **Y** from **X** and $\mathbf{S}^{(n)}$	Improve predictions
Baseline model	$f^{Y\mid\mathbf{X}}$	Predicts **Y** from **X** alone	Verify improvement
Environment-oracle model	$f^{Y\mid\mathbf{X},E}$	Predicts **Y** from **X** and E	Verify Criterion 3
Contextual environment model	$f^{E\mid\mathbf{X},\mathbf{S}^{(n)}}$	Predicts E from **X** and $\mathbf{S}^{(n)}$	Verify Criterion 2
Baseline environment model	$f^{E\mid\mathbf{X}}$	Predicts E from **X** alone	Verify Criterion 2.

Table 6.1. The five different model types used to evaluate our approach and verify the theoretical criteria for improvement.

data, where observations are organized into clusters or levels [222, 223], mirroring the notions of domain or environment in DG. Indeed, hierarchical Bayesian models have been successfully applied in many areas of science, but are typically constrained to linear or generalized linear models (GLMs), prioritizing interpretability over predictive performance.

From a somewhat different Bayesian perspective, IID data represents the starting point for learning invariant summary statistics for parameter estimation or model comparison. The pioneering work on *Neural Statisticians* [224] tackles the task from a variational perspective, learning global set-representations as part of a generative model. Neural processes [225, 226] comprise a related family of set-based models for prediction and uncertainty quantification in supervised learning tasks, rooted in the spirit of Gaussian processes [227]. Finally, [228] explore a permutation-invariant variational autoencoder (SetVAE) with multiple latent variables trained with a modified ELBO objective. The goal of this and follow-up models [229] is accurate generative performance, less so the learning of compact representations. The theoretical and empirical implications of learning random set-representations falls outside the scope of our current work, but offers a potentially fruitful avenue for future research.

Crucially, none of the above methods pursues the concrete goal of our work, which is improving domain generalization performance through set-based environment representations. For the purpose of achieving this goal, we employ variants of the `DeepSet` [209] and `SetTransformer` architectures for our backbone set-encoder throughout all experiments. Interestingly, we observe that our approach is widely robust to the choice of set-encoder architectures in the problems considered.

6.7.3. OOD Detection and Selective Classification

Detecting unusual inputs that deviate from the examples in the training set has been a long-standing problem of conceptual complexity in machine and statistical learning [6, 37, 230–232]. Flagging out-of-distribution (OOD) instances involves identifying uncommon data points that might compromise the reliability of machine learning systems [37]. OOD detection is closely related to *inference with a reject option* (also termed selective classification) [35, 36], which allows classifiers to refrain from making a prediction under ambiguous or novel conditions [233]. The reject option has been extensively studied in statistical and machine learning [234–237], with early work dating back to the 1950s[234, 238, 239].

More recently, our work in [41] explored the utility of selective classification in DG settings. They investigated various *post-hoc scores* to define a “competence region” in feature space where a classifier is deemed competent. Post-hoc methods rely on various as-

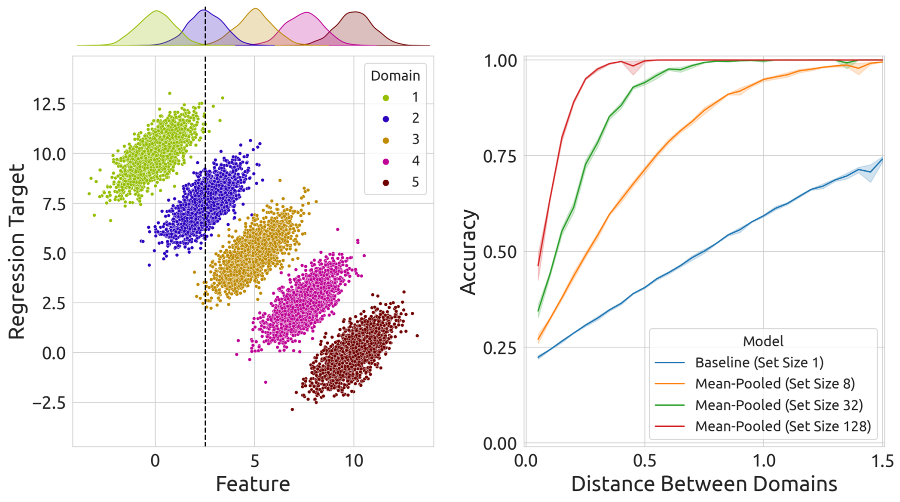


Figure 6.6. Experiment 1. *Left*: Toy dataset that conforms to our theoretical criteria. A detailed description can be found in Appendix B.3.1. Without environmental information, the marked input at $x = 2.5$ could belong to either one of the domains numbered 1, 2, or 3. Marginal distributions of the five environments are shown on top. *Right*: Comparison of environment classification accuracy for different set sizes.

pects of pre-trained model outputs, such as the softmax outputs (e.g., [240]), logit outputs (e.g., [241, 242]), or intermediate features (e.g., [243–245]). In this work, we consider a post-hoc score based on the k-nearest neighbors to the training set in feature space similar to [244], which is applicable to both classification and regression settings. Unlike the approach taken in our work in [41], where the focus lies on individual input features, we consider the set summary provided by the set-encoder. Thus, we can identify novel environments even when singleton inputs lack sufficient information.

6.8. Experiments

In the following, we explore various aspects of our approach across three different dimensions. First, we show on two datasets that our model achieves improved performance in the ID as well as the OOD setting compared to a baseline model. Second, we show how novel environments can be detected to select between the most predictive (in the ID setting) and the most robust (in the OOD setting) model. We also show that novel environment detection can be utilized to avoid failure cases. Third, we demonstrate that the necessary criteria (see Subsection 6.6.5) can be validated empirically, identifying cases where no benefits of our method can be expected. Experimental details can be found in the Appendix B.

6.8.1. Evaluation Approach

To approximate Criterion 1, Criterion 2 and Criterion 3, we are required to train five distinct models (see Table 6.1 for an overview). We denote our *composite model* as $f^{Y|\mathbf{X},\mathbf{S}^{(n)}}$ (see Figure 6.5) and the *baseline model* (having no access to the context) as $f^{Y|\mathbf{X}}$. Based on these two models, we can compute the *relative improvement* achieved by our model relative to a baseline model via

$$\mathcal{R}_{\text{I}} = \frac{\mathcal{L}(f^{Y|\mathbf{x},\mathbf{s}^{(n)}}) - \mathcal{L}(f^{Y|\mathbf{x}})}{\mathcal{L}(f^{Y|\mathbf{x}})}. \tag{6.24}$$

Here, $\mathcal{L}(f^{Y|\mathbf{x},\mathbf{s}^{(n)}})$ denotes a performance measure for our model and similarly for the baseline model $f^{Y|\mathbf{x}}$. $\mathcal{R}_{\text{I}} > 0$ signifies an advantage attained by our approach and therefore the fulfillment of Criterion 1. In the regression setting, we consider the negative L2-Loss as the performance measure.

To validate Criterion 2, we train a *contextual environment model* (referred to as $f^{E|\mathbf{x},\mathbf{s}^{(n)}}$) utilizing both the set input $\mathbf{s}^{(n)}$ and the target input $\mathbf{x}$ to predict the environment label E. Additionally, we train a *baseline environment model* (denoted as $f^{E|\mathbf{x}}$) aimed at predicting E solely from $\mathbf{x}$. We then compute the relative improvement $\mathcal{R}_{\text{II}}$ of the contextual environment predictor relative to the baseline environment predictor:

$$\mathcal{R}_{\text{II}} = \frac{\text{Acc}(f^{E|\mathbf{x},\mathbf{s}^{(n)}}) - \text{Acc}(f^{E|\mathbf{x}})}{\text{Acc}(f^{E|\mathbf{x}})} \tag{6.25}$$

We consider here the accuracy of the environment prediction as a performance measure. $\mathcal{R}_{\text{II}} > 0$ indicates that Criterion 2 is satisfied. In our experiments, we choose the set size n such that we achieve approximately 100% accuracy for our contextual environment predictor $f^{E|\mathbf{x},\mathbf{s}^{(n)}}$ on ID data.

Similarly, we consider an *environment-oracle model* $f^{Y|\mathbf{x},E}$ that aims to predict Y from the singleton input and the environment label E. We define the relative improvement $\mathcal{R}_{\text{III}}$ of the environment-oracle model $f^{Y|\mathbf{x},E}$ compared to the baseline method $f^{Y|\mathbf{x}}$:

$$\mathcal{R}_{\text{III}} = \frac{\mathcal{L}(f^{Y|\mathbf{x},E}) - \mathcal{L}(f^{Y|\mathbf{x}})}{\mathcal{L}(f^{Y|\mathbf{x}})} \tag{6.26}$$

In this case, the relative improvement $\mathcal{R}_{\text{III}}$ is associated with Criterion 3.

6.8.2. Experiment 1: Toy Example

Setup To set the stage, we consider a dataset that is inspired by [246]. The dataset includes data from five environments, defined by five 2D Gaussian distributions differing only in their locations (i.e., mean vectors). The data-generating process is thus:

$$j \sim \text{Categorical}(p_1, \dots, p_J) \tag{6.27}$$

$$\mathbf{x}^{(ij)} \sim \mathcal{N}(\boldsymbol{\mu}^{(j)}, \boldsymbol{\Sigma}) \quad \text{for} \quad i = 1, \dots, I_j \tag{6.28}$$

This is the classical setting of Simpsons' paradox, for which a naive linear model fit oblivious to the *hierarchical* data-generating process will yield results opposite to the true relationship between $X_1^{(j)}$ and $X_2^{(j)}$ in each environment (cf. Figure 6.6 and Appendix B.3.1 for details).

We designate the first dimension, X_1, as the input feature, while the second dimension, $Y = X_2$, serves as the outcome variable. Importantly, the dataset meets our necessary criteria: We cannot infer the origin environment from a single input alone, as indicated by the overlap between the marginal distributions of X_1, obtained by projecting the entire

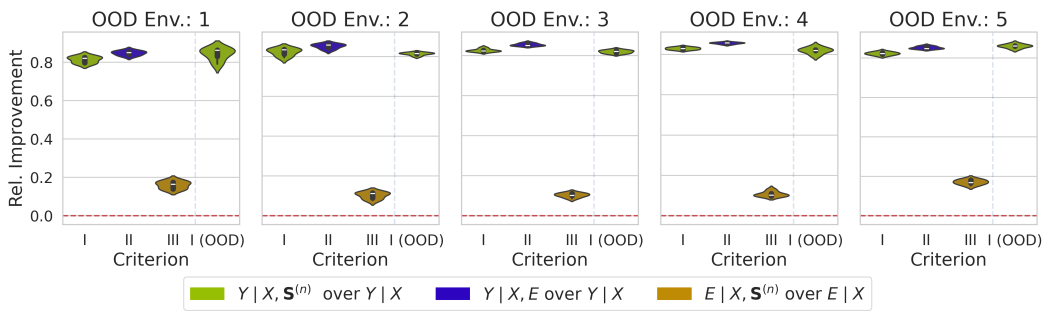


Figure 6.7. Experiment 1. Relative improvement of set-encoder (shown in I) approach versus baseline model (0 means no improvement is achieved) on toy example. We also show I (OOD) on OOD data. II depicts the relative improvement of the environment-oracle model compared to the baseline model. III demonstrates the relative improvement in predicting the environment when using contextual information compared to the absence of it. Sampling variation arises from using different seeds to partition the ID data into training, test and validation set.

sample onto the x-axis in Figure 6.6. Thus, this setting aligns with Criterion 2, and, additionally, corresponds to Criterion 3, since learning about the environment location $\boldsymbol{\mu}^{(j)}$ should improve prediction.

Results As a first check of Criterion 2, we evaluate whether a set input provides additional information about the environment compared to a singleton input. This evaluation involves studying the classification accuracy in distinguishing between the environments. In Figure 6.6 we observe that the additional set input improves the ability to predict the environment significantly and the more samples we include, the better the prediction. As expected, a decrease in the distance between environments necessitates more samples to differentiate between environments. Interestingly, the particular choice of architecture for the permutation-invariant network does not seem to play a significant role for predicting the environment label well, as demonstrated in Section B.7.

Next, we assess the predictive capabilities of our approach across all possible scenarios of "leave-one-environment-out". This involves training on all environments except one and treating the excluded environment as a novel OOD scenario. Here, we consider linear models to ensure an optimal inductive bias for the problem (non-linear models achieve similar results, as shown in Appendix B.3.3). We can see that Criterion 1, Criterion 2 and Criterion 3 are satisfied in Figure 6.7. Providing contextual information in the form of a set input increases the performance significantly compared to a baseline model in the ID as well as in the OOD setting (see I and I (OOD) in Figure 6.7). We also observe a slightly higher relative improvement when the environment label is directly provided (see II) compared to using the output of the set-encoder (see I). This aligns with our expectations, as the set input does not offer more information about the target value than the environment label itself.

Finally, we visualize the predictions of the baseline approach and our set-encoder approach in Figure 6.8. Our model captures and utilizes the characteristics of each environment for prediction. In contrast, the baseline approach struggles to discern between environments due to the significant overlap between environments, resulting in an inability to deal with environmental differences. Note that we obtained the best results on this problem by considering a class of linear models that aligns with the data-generating process.

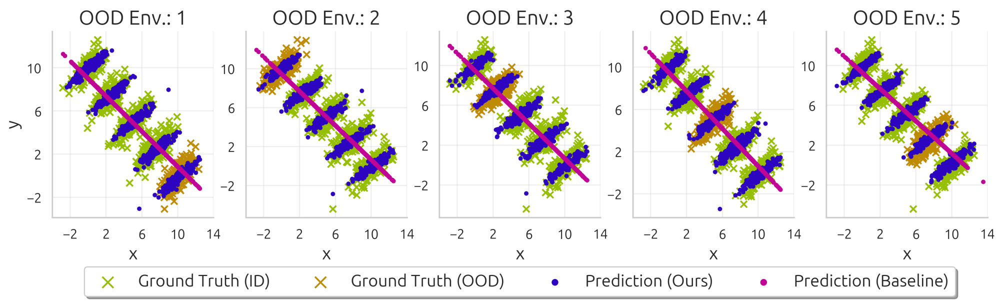


Figure 6.8. Experiment 1. Predictions performed on the toy dataset illustrated in Figure 6.6. We show predictions made by both our set-encoder approach and the vanilla model in the ID and OOD settings.

However, we observe that extrapolation performance drops when the considered models are overly complex and lack a strong inductive bias (see Appendix B.3.3).

6.8.3. Experiment 2: ProDAS Example

Setup We utilize the ProDAS library introduced in Section 6.1 to generate high-dimensional image data that meets our dataset requirements. The dataset comprises objects of shape square and circle, exhibiting variations in their texture, background color, rotation, and size. Additionally, the background varies in color and texture, resulting in a complex scenario. We consider the task of predicting the object size. Difficulties arise due to the presence of distinct environments with varying characteristics. Specifically, depending on the environment, a constant is added to the observed object size to get the actual target variable that we aim to predict:

$$Y_{\text{gt}} = Y_{\text{observed}} + j \cdot \text{const}_1 \tag{6.29}$$

Here, $j \in \{1, 2, 3, 4\}$ denotes the environment, while Y_{gt} represents the ground truth (or factual) size, obtained as a sum of the observed size Y_{observed} (relative to the image frame) and a constant depending on j. The background color follows a normal distribution $\mathcal{N}(\boldsymbol{\mu}_j; \boldsymbol{\Sigma})$ where the mean depends on the environment in the following way: $\boldsymbol{\mu}_j = \boldsymbol{\mu}_0 + j \cdot \text{const}_2$. Here we assign a small value to const_2 to enforce the background distributions to overlap between different environments. Specifically, this construction implies that the relation between input $\mathbf{X}$ and target Y differs across environments. This corresponds to Criterion 3. Notably, inferring the originating environment from a single sample is unattainable due to overlapping background distributions (corresponding to Criterion 2). Samples of different environments are shown in Section B.4. This example could be inspired by microscopy data where different microscopes correspond to distinct environments, each exhibiting its own characteristics. During training, we assume to have access to the ground truth value Y_{gt}.

Results In line with the results from the previous toy example, we can demonstrate a strong relative improvement in the ProDAS dataset, as depicted in Figure 6.9. All formal criteria are satisfied and a very significant improvement is achieved, both in the ID and the OOD setting, by considering the contextual information from the environment. Details for this experiment can be found in Section B.4.

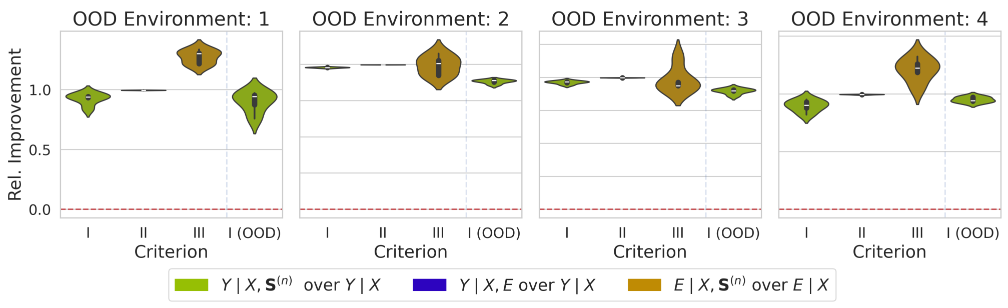


Figure 6.9. Experiment 2: Relative improvement of set-encoder (shown in I) approach versus baseline model (0 means, no improvement is achieved) on ProDAS dataset. We also show I (OOD) on OOD data. II depicts the relative improvement of the environment-oracle model compared to the baseline model. III demonstrates the relative improvement in predicting the environment when using contextual information compared to the absence of it. Variations arise from using different seeds to partition the ID data into training, test and validation set.

6.8.4. Experiment 3: Colored MNIST

Setup The ColoredMNIST dataset (see also Figure 2.1) is an extension of the standard MNIST dataset, wherein the number of classes is reduced to two classes (all standard labels < 5 are assigned to new label 0, and all labels ≥ 5 are the new label 1). Furthermore, there exists label noise, so only in 75% of all cases, the label can correctly be predicted from the input image. To make things more challenging, the image background can take two colors that are also associated with the image label. In the first environment, the association is 90% and in the second one 80%. Therefore, a baseline model would tend to utilize the background for prediction instead of the actual shape. However, in a third environment, the associations are reversed, so that a model based on the background color would achieve only 10% accuracy – worse than random.

This dataset implies a trade-off between predictive performance in ID domains versus robustness in OOD domains, as discussed in Subsection 5.5.2. For instance, an invariant model that relies solely on an object's shape would be robust to domain shift at the cost of diminished accuracy in the first two environments (75% instead of 80% or 90%). In contrast, a baseline model would achieve greater accuracy in the first domains (80% and 90%), but would fail dramatically in the third domain (only 10%).

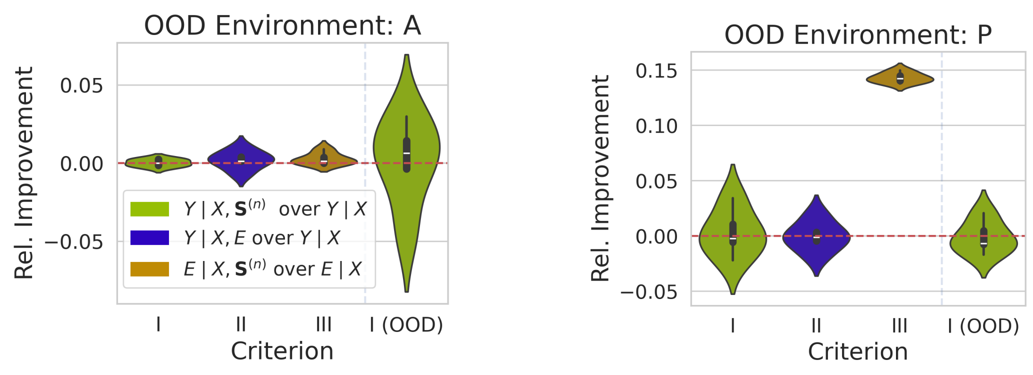
Results Here, we assume the invariant model to be given (see Section B.5 for details), but it could also be obtained by invariant learning, e.g. Invariant Risk Minimization [34]. With our novel environment detection approach (see Subsection 6.6.7) we can get the best of both worlds, circumventing the inherent trade-off. When identifying the ID setting, we utilize the baseline model that achieves the highest predictiveness within the observed environments. In case we detect the OOD setting, we employ the invariant model. We compare this kind of model selection due to the features $h_\psi(\mathbf{S}^{(n)})$ inherent to our model versus the features extracted by the baseline model. The results can be found in Table 6.2. By utilizing the model selection based on the set-summary $h_\psi(\mathbf{S}^{(n)})$, we nearly recover the ID accuracy while maintaining identical performance to the invariant model on OOD data. Evidently, the novel environment detection only works with set summaries. A feature extracted from a single sample does not provide enough information to reliably detect distribution shifts, leading to difficulties in effectively selecting between baseline and invariant model, as demonstrated in Table 6.2.

	Accuracy [%] ↑	
	In-Distribution	Out-of-Distribution
Baseline	**84.6** $\pm$ 0.3	10.2 $\pm$ 0.3
Invariant	72.8 $\pm$ 0.9	**73.1** $\pm$ 0.2
Selection (Ours)	**84.1** $\pm$ 0.3	**73.1** $\pm$ 0.2
Selection (Baseline)	**84.0** $\pm$ 0.3	14.0 $\pm$ 0.4
Optimal Classifier	**85.0**	**75.0**

Table 6.2. Experiment 3. Mean and standard deviation of accuracy percentages across model types and domain settings over 5 runs. The features extracted by our model allow for improved OOD detection compared to the features of the baseline model. Thus, our model can perform a favorable selection between the baseline model in the ID setting and the invariant model in the OOD setting.

6.8.5. Experiment 4: Violated Criteria

Setup We consider the PACS dataset [46], training our model on the Cartoons, Sketches, and Paintings environments, and assess its performance in the Art environment during testing. The dataset includes images with labels that we intend to predict. Regarding a second classification task, we delve into the OfficeHome dataset [48]. In line with the PACS dataset, we approach the classification problem, training across three specific environments, and subsequently evaluating a novel one as an out-of-distribution (OOD) scenario.



(a) Environment *Art* in PACS dataset. The environment is almost completely inferable from one input sample (Criterion 2 not satisfied). Conclusively our approach does not yield benefits.

(b) Environment *Product* in OfficeHome dataset. Although the environment is not inferable from one input sample (Criterion 2), the environment information does not yield benefits (Criterion 3).

Figure 6.10. Experiment 4. Examples where at least one of the necessary criteria is not satisfied and our approach cannot possibly yield benefits. For experimental details, see Section B.6.

Results When the criteria are not met, no benefits can be achieved, even in the ID setting. This has been proven in Theorem 2 and we demonstrate it here for two scenarios empirically (see Figure 6.10). We find that Criterion 2 is not satisfied on the PACS dataset: As depicted in Figure 6.10b, the contextual environment model $f^{E|\mathbf{X},\mathbf{S}^{(n)}}$ does not perform

Spring

	MSE ↓		AUROC [%] ↑
	ID	OOD	
Baseline	2.89 ± 0.15	$\mathbf{2.94} \pm 0.05$	50.8 ± 2.2
Ours	$\mathbf{2.13} \pm 0.13$	3.23 ± 0.11	$\mathbf{99.7} \pm 0.2$

Summer

	MSE ↓		AUROC [%] ↑
	ID	OOD	
Baseline	2.99 ± 0.17	$\mathbf{2.74} \pm 0.10$	65 ± 5
Ours	$\mathbf{2.27} \pm 0.13$	3.8 ± 0.4	$\mathbf{100.0} \pm 0.0$

Fall

	MSE ↓		AUROC [%] ↑
	ID	OOD	
Baseline	2.29 ± 0.12	$\mathbf{7.0} \pm 0.4$	76.4 ± 2.6
Ours	$\mathbf{2.19} \pm 0.09$	14.90 ± 1.30	$\mathbf{100.0} \pm 0.0$

Winter

	MSE ↓		AUROC [%] ↑
	ID	OOD	
Baseline	2.21 ± 0.11	6.08 ± 0.13	58.2 ± 0.7
Ours	$\mathbf{2.09} \pm 0.12$	$\mathbf{5.7} \pm 0.4$	$\mathbf{100.0} \pm 0.0$

Table 6.3. Experiment 5. Performance comparison between our model and the baseline, broken down by target domain. We compare the performance in the ID and OOD setting (MSE), as well as their capability to detect a novel environment (AUROC). Both models fail in the OOD setting, but our model can detect with strong certainty when this is the case. We present the mean and standard deviation derived from 5 runs using different seeds for partitioning into training, validation, and test sets.

better compared to the baseline environment model $f^{E|\mathbf{x}}$. Remarkably, a single example is sufficient to infer the source environment, allowing for a 99.7% accuracy in predicting the correct environment from an individual sample (see Section B.6). Since Criterion 2 is not fulfilled, we anticipate Criterion 3 also to be wrong. This is indeed the case as Figure 6.10b depicts. Since the criteria are not met, we do not achieve any benefit over the baseline model, neither in the ID nor in the OOD setting, as demonstrated in Figure 6.10b.

On the OfficeHome dataset we find that Criterion 2 is not satisfied, while Criterion 3 is. Results are depicted in Figure 6.10b. We observe that the set input offers benefits for predicting the data originating environment corresponding to Criterion 3. However, even when providing the target classifier with the environment label (environment-oracle model), we do not achieve an improvement over the baseline model suggesting that Criterion 2 is not satisfied. As expected and depicted in Figure 6.10b, our method does not yield benefits compared to the baseline model.

6.8.6. Experiment 5: Failure Case Detection

Setup Besides unfulfilled criteria, another reason why our approach might fail to reap benefits or degrade in performance is when the distribution shift requires extrapolation. This might be unattainable by the model. We demonstrate, using the BikeSharing dataset [247], that in cases where different seasons like summer or winter represent distinct environments, extrapolation might be necessary. For this dataset we consider the task of predicting the number of bikes rented across the day based on weather data. Details about the dataset and pre-processing steps can be found in Section B.8. We explore four scenarios, each entailing training on all seasons except one. We aim to assess the abilities of our model compared to a baseline model in detecting novel environments.

Results In Table 6.3 we demonstrate that our approach is slightly superior compared to the baseline model in the ID settings. However, both the baseline and our approach experience performance degradation in the novel environments. To detect the novel environment (and consequentially potential failure cases), we compute the score as suggested in Subsection 6.6.7 and evaluate how well it distinguishes between ID versus OOD environment. We designate an independent ID test set and use the environment excluded during training (e.g., summer or winter) as the OOD set for evaluation. The area under the ROC-curve (AUROC) in Table 6.3 demonstrates that the score based on the summary vector provided by the permutation invariant network allows for a perfect novel environment detection whereas the standard approach fails in detecting the novel environment.

6.9. Conlusion: Context-Aware DG

In this chapter, we introduce a novel approach enabling the extraction and utilization of contextual information from set inputs within a Domain Generalization (DG) setting. The set inputs originate from the same environment as the sample for which a prediction is required and are summarized into a vector via a permutation-invariant network. We formulate criteria that are necessary for our approach to yield benefits and are easy to verify. Empirically, we demonstrate that we can verify these criteria, which enables us to identify cases where our approach does not yield benefits. We showcase the merits of our approach on several datasets. Additionally, we demonstrate that novel environments can be detected, allowing for the identification of potential failure cases.

In our current framing, we have not yet employed regularization techniques, such as enforcing the set vector to contain all information about the environment in our composite model in Figure 6.5. In general, investigating inductive biases for the composite network architecture is an interesting future research avenue. Finally, even though we focused our experiments in standard supervised learning settings, our method could also be employed in other realms, such as domain disentanglement, data generation, or in combination with other DG methods. Thus, extending our method to different settings represents another exciting research direction.

Finding Competence Regions in Domain Generalization 7

The following is a direct adaptation from our work in [41].

Abstract We investigate a "learning to reject" framework to address the problem of silent failures in Domain Generalization (DG), where the test distribution differs from the training distribution. Assuming a mild distribution shift, we wish to accept out-of-distribution (OOD) data from a new domain whenever a model's estimated competence foresees trustworthy responses, instead of rejecting OOD data outright. Trustworthiness is then predicted via a proxy *incompetence score* that is tightly linked to the performance of a classifier. We present a comprehensive experimental evaluation of existing proxy scores as incompetence scores for classification and highlight the resulting trade-offs between rejection rate and accuracy gain. For comparability with prior work, we focus on standard DG benchmarks and consider the effect of measuring incompetence via different learned representations in a closed versus an open world setting. Our results suggest that increasing incompetence scores are indeed predictive of reduced accuracy, leading to significant improvements of the average accuracy below a suitable incompetence threshold. However, the scores are not yet good enough to allow for a favorable accuracy/rejection trade-off in all tested domains. Surprisingly, our results also indicate that classifiers optimized for DG robustness do not outperform a naive Empirical Risk Minimization (ERM) baseline in the competence region, that is, where test samples elicit low incompetence scores.

7.1. Introduction

Although modern deep learning methods exhibit excellent generalization, they are prone to silent failures when the actual data distribution differs from the distribution during training [37, 248]. We address this problem in a "learning to reject" framework [233, 249, 250]: *Given a pre-trained model and potentially problematic data instances, can we determine if the model's responses are still trustworthy?*

A major goal of this work is to explore the above question in settings where we wish to make predictions on a test set from a *new domain* following a potentially different distribution than the one available during training. This setting is referred to as Domain Generalization (DG) and assumes that we have access to multiple domains (also known as datasets or environments) during training (for a thorough introduction to the topic see Section 2.2).

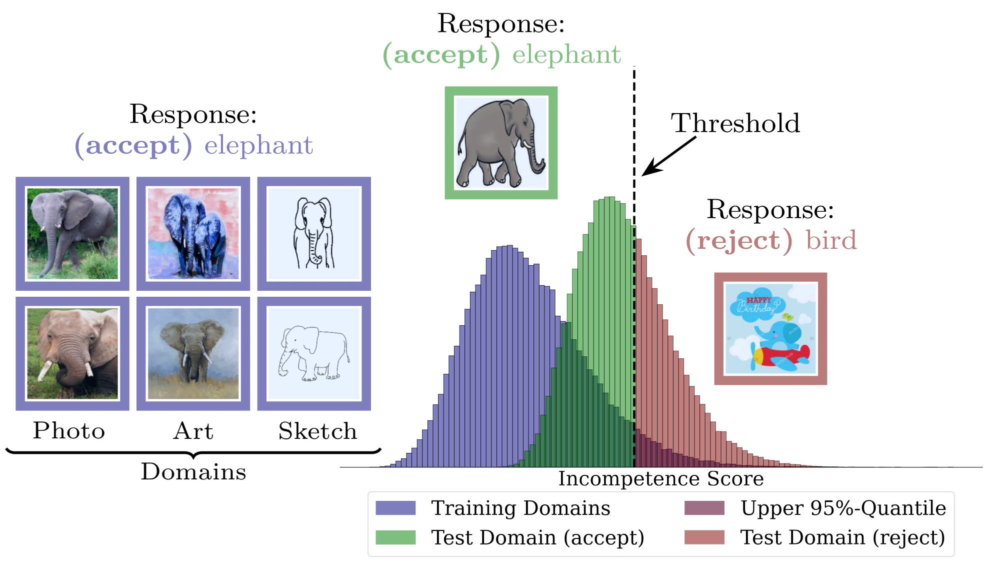


Figure 7.1. The main principle behind *incompetence scores* for improved domain generalization: We reject instances above the incompetence threshold, which is located at the 95% quantile of the training distribution.

The generalization task asks to provide accurate predictions for a new domain, usually subject to a mild distribution shift (e.g., from one hospital to the next). Still, from a data-centric perspective, almost all instances in the new domain are out-of-distribution (OOD). Following the rationale of DG, we do not want to reject all OOD instances outright, but only those for which the estimated model competence falls below some acceptance threshold.

Since we do not have access to the distribution of the test data during training, we can neither determine out-of-domain competence directly [251], nor define the acceptance threshold in a Bayes-optimal way [239]. Instead, we investigate proxy scores that are negatively correlated with competence: We call them *incompetence scores* and they should monotonically decrease as a model's accuracy increases (see Section 7.3). For a simple example of such a score, we may consider the distance of a new data point to the nearest neighbor of the training data in a model's learned feature space. In this case, we expect the performance to drop with increasing distance. A visual explanation of the incompetence score is shown in Figure 7.1. Interestingly, our experiments demonstrate that the monotonicity property typically holds for well-known choices of these scores.

Setting a threshold to delineate a *competence region* inevitably results in a trade-off between accuracy and coverage: The more instances we add to the competence region, the worse the accuracy, and *vice versa*. Identifying the optimal (task-dependent) trade-off in DG is difficult due to the differences between the training and the (unknown) test distributions. Thus, we find it pertinent to explore this trade-off for different thresholds across DG tasks (see Subsection 7.4.3).

The concept of incompetence underlying the present work is strongly linked to previous research on classification with a reject option (e.g. [252]) and selective classification (e.g. [35]). Common to these concepts is the idea to accurately predict errors based on a proxy quantity. Since we are interested in the task-dependent competence of pre-trained classifiers, we concentrate solely on post-hoc OOD detection methods as proxy scores for incompetence. And although the current work focuses on classification tasks, our approach should also be worth pursuing in regression tasks, since feature-based incompetence scores seem equally applicable in this case.

In the following, we present a comprehensive experimental evaluation of incompetence scores in a variety of DG tasks. For comparability with prior work, we focus on standard datasets from the DG literature [51] and consider the closed vs. open world setting (i.e., new appearances of known classes vs. hitherto unknown classes) as well as the effect of measuring incompetence through different data representations. Further, we investigate whether state-of-the-art classifiers that are optimized specifically for domain shift robustness exhibit more accurate competence regions than naively trained ones. Finally, we investigate whether it is possible to estimate an incompetence threshold, such that a classifier is guaranteed to recover its ID accuracy in the corresponding competence region under domain shift. In summary, we make the following contributions:

1. We demonstrate empirically that accuracy decreases as incompetence scores increase and highlight the resulting trade-offs between rejection rate and accuracy gain (see Subsection 7.4.3)
2. We find that both feature- and logit-based scores are competitive in the closed world, whereas feature-based approaches work best in the open world setting (see Subsection 7.4.4 and Subsection 7.4.5)
3. We propose an approach to determine an incompetence threshold from ID data and demonstrate its utility for most domain shifts considered in this work (see Subsection 7.4.6)
4. We observe that robust classifiers do not outperform a naive baseline in terms of generalization performance in the elicited competence regions (see Subsection 7.4.4)

7.2. Related Work

7.2.1. OOD Detection

Dealing with anomalous (i.e., out-of-distribution; OOD) instances that differ from those contained in the training set (i.e., our proxy for the in-distribution; ID) is a widely discussed and conceptually overloaded topic in the machine and statistical learning literature [6, 37, 230–232]. OOD detection addresses the problem of flagging unusual data points which could undermine the reliability of machine learning systems [37]; OOD generalization addresses the need to make predictions even when the test distribution is completely unknown or known to be different than the training distribution [6].

In this work, we are interested in analyzing established domain-robust classifiers. Thus, we focus on OOD detection methods that do not modify the classifier architecture or training. Such methods are called *post-hoc* detection [37], as they do not intervene on the downstream classifier. In this work, we utilize established post-hoc methods that rely on various aspects of model output such as the softmax output (e.g. [240]), logit output (e.g. [241, 242]), or intermediate feature-outputs (e.g. [243–245]).

Post-hoc OOD scores have been shown to perform well across a variety of OOD detection benchmarks [232]. Previous work analyzed post-hoc OOD detection scores to predict the accuracy of a classifier on novel inputs [253] or to detect ID failure cases [254].

In addition, [253] compute an aggregated OOD-score over an entire ID dataset to predict the global accuracy of a classifier on OOD data. Differently, we aim to predict the likelihood of error from individual incompetence score values and show that this approach provides us with a finer control over the trade-off between coverage and accuracy (see Subsection 7.4.6).

Despite the large volume of literature focusing on OOD detection and generalization (e.g., [255]), there are no extensive studies applying OOD scores to domain generalization (DG) benchmarks. Thus, one of the main goals of this work was to provide such a comprehensive analysis on the utility of OOD scores for improving DG.

7.2.2. Domain Generalization

The goal of domain generalization (DG) is to train models that generalize well under distribution shifts [15, 17], such as adversarial attacks [44] or style changes [256], for which the label space remains unchanged during testing [37]. In DG settings, we assume that we have access to different environments or datasets (e.g., art and sketch images) and the goal is to make good predictions in completely unknown environments (e.g., real-world images). We introduce DG in Section 2.2 in more depth.

Compared to Domain Adaptation [DA; 52], where we have unlabeled data from the test domain, the DG problems assume that we have no knowledge about the test domain(s). Consequently, it is not possible to train the algorithm using unlabeled test data as in self-training [257]. However, a recent study has demonstrated that a model can be effectively adjusted during test time [258]. Moreover, it has been shown that classifiers can assign high likelihoods under domain shift even when they are plainly wrong, which makes it hard to detect failure cases [259, 260]. Thus, proxy “incompetence” OOD scores appear to be good candidates for spotlighting such failures. However, to the best of our knowledge, there are no extensive studies which attempt to quantify the competence of domain-robust models in the context of DG.

Many benchmark datasets in DG have been established, on which researchers can study generalization performance beyond a single training environment [14, 51]. In this work, we consider the main datasets contained in the DomainBed benchmark [51]. We additionally distinguish between a *closed world* setting, where only instances of known classes are encountered in the test domain, and an *open world* setting, where instances of unknown classes are also present in the test domain. We believe the open world setting to be of practical interest, even though typical DG problems are formulated under a closed world assumption [17].

The current work primarily focuses on learning-based approaches that seek to learn features that remain invariant under domain shift. For an overview of other approaches see for instance Subsection 2.2.3. According to [261], there is theoretical evidence to suggest that features that remain invariant across domains enable accurate predictions in cases of distributional shifts. As a result, various algorithms have been proposed with the goal of learning invariant features [15, 31, 34, 262]. However, it is not clear which DG methods can achieve consistently robust performance across different datasets. On the one hand, it has been suggested that a strong standard classifier trained with empirical risk estimation (ERM) performs favorably across multiple DG datasets [51, 263]. On the other hand, some DG methods have been shown to outperform an ERM baseline on several benchmark datasets [14]. Here, we complement the existing literature by examining whether the competence regions of different DG classifiers differ in terms of the achieved improvements in accuracy.

7.2.3. Selective Classification

Inference with a reject option [aka *selective classification*, 35, 36] enables classifiers to refrain from making a prediction under ambiguous or novel conditions [233]. The reject option has been extensively studied in statistical and machine learning [234–237]. The origins of these approaches can be traced back until at least the 50s of the last century, as demonstrated by works such as [234, 238, 239]. However, selective classification has only recently gained attention in the context of deep neural networks [35].

[250] outline the three main reasons why a reject option could be a reasonable choice in any practical application: 1) failure cases; 2) unknown cases; and 3) fake inputs. For instance, [264] train natural language processing (NLP) models for selective question answering under domain shift. [265] investigate the utility of MaxProb (a common OOD detection score) as a rejection criterion across several NLP datasets. [266] use the Mahalanobis distance as OOD detection method to filter inputs to NLP models for conditional text generation and [267] showcase the reject option for catching software defects.

The main challenge selective classifiers face is how to reduce the error rate by "rejecting" instances for which no reliable prediction can be made, while keeping coverage (i.e., the number of "accepted" instances) as high as possible [239, 268, 269]. And while the theoretical characteristics of the resulting trade-off have been systematically studied [36, 270], the empirical utility of OOD "rejection scores" for ensuring robust performance in the DG setting remains largely unclear. In this work, we perform an extensive evaluation of this trade-off across a wide variety of state-of-the-art OOD scores, domain-robust classifiers, DG datasets and environments.

7.3. Method

We denote with f_θ an arbitrary classifier with a vector of trainable parameters θ (e.g., neural network weights) which we typically suppress for readability. To evaluate a classifier, we consider its accuracy, which we denote as A_{dist}, based on inference queries from some reference distribution $x \sim p_{\text{dist}}(x)$.

7.3.1. Incompetence Scores

The goal of an incompetence score $s_f \colon \mathbb{R}^D \to \mathbb{R}$ is to indicate whether a classifier f is familiar with some input $x \in \mathcal{X}$. We consider familiarity with the input to be equivalent to competence. The fundamental principle of this work is that instances eliciting a high incompetence score are intrinsically hard to predict and *vice versa*. Due to the close conceptual connection between competence and familiarity or incompetence and OOD, we employ OOD scores as proxy for incompetence. In particular, we employ *post-hoc* methods that compute an OOD score taking into account the classifier.

In our subsequent experiments, we compute the incompetence scores via a number of *post-hoc* methods. The *post-hoc* methods used in this chapter can be grouped into the following categories:

- Feature-based: Virtual-logit Matching [ViM; 245], Deep-KNN [Deep-KNN; 244];
- Density-based: Gaussian mixture models (GMM), minimum Mahalanobis distance between features and class-wise centroids [Mahalanobis; 271];
- Reconstruction-based: reconstruction error of PCA in feature space [243];
- Logit-based: energy score [Energy; 242], maximum logit [Logit; 241], maximum softmax [Softmax; 240], and energy-react [Energy-React; 272].

Note, that we interpret higher scores as indicative of incompetence (e.g., we consider the negative of the maximum softmax and the maximum logit).

7.3.2. Admissible Incompetence Scores

Detecting out-of-competence means checking whether some given incompetence score $s_f(x) \in \mathbb{R}$ falls below some threshold α (classified as in-competence) or above (classified as out-of-competence). We consider scores $s_f(x)$ that depend on the classifier f and the input x at hand.

The threshold α trades off accuracy (how well does the classifier perform on accepted data) with coverage (how many samples does the score accept). In this section, we describe how a useful (ideal) incompetence score should affect downstream classification as a function of the threshold α. In particular, consider the subset of input space where the classifier is deemed competent given a fixed threshold α:

$$X_f(\alpha) := \{x : s_f(x) \leq \alpha\}. \tag{7.1}$$

We use the ID data to determine a suitable threshold for the competence region, for instance we later pick $\alpha = \alpha_{95}$ such that 95% of the ID data is in $X_f(\alpha_{95})$. We consider the accuracy $\mathrm{A}_{\mathrm{OOD}}(\alpha)$ of the classifier c on the unknown test domain restricted to the competence region $X_f(\alpha)$ as a function of α.

We summarize the above description in the fundamental criterion of this work: **An admissible incompetence score must assign low incompetence to those regions where the downstream accuracy is high.** We formalize this as follows:

Criterion 4. *An incompetence score $s_f(x)$ is called "admissible" if the downstream accuracy $\mathrm{A}_{\mathrm{OOD}}(\alpha)$ decreases monotonically as α is increased for any distribution that undergoes a mild distribution shift.*

This monotonic trend requires that the incompetence score $s_f(x)$ is closely related to the performance of the classifier. Such a connection allows us to make predictions on the downstream accuracy as a function of α:

Proposition 8. *Given a classifier $f_\theta(x)$ and its corresponding in-distribution p_{ID}. Then, for a test distribution of interest p_{OOD} and a corresponding admissible score $s_f(x)$ as in Criterion 4:*

(a) If there is a threshold $\alpha^* \in \mathbb{R}$ such that for all $\alpha \leq \alpha^*$ ID and OOD have the same support and classification accuracy, then, $\mathrm{A}_{\text{OOD}}(\alpha) \geq \mathrm{A}_{\text{ID}}$ for $\alpha < \alpha^*$.

(b) In the limit of $\alpha \to \infty$, we find that $\mathrm{A}_{\text{OOD}}(\alpha) \to \mathrm{A}_{\text{OOD}}$.

Proof. (b) Take the limit $X_f(\alpha) \xrightarrow{\alpha\to\infty} \mathbb{R}^D$. Then there is no restriction of the support of p_{OOD}, so the accuracy for large α approaches the accuracy on the entire OOD dataset.

(a) By assumption p_{ID} and p_{OOD} share their support when restricted to the competence region $X_f(\alpha)$ when $\alpha \leq \alpha^*$. Thus we can always assume that $P_{\text{OOD}}(X_f(\alpha)) > 0$ for all $\alpha \geq \min_{x\in\text{supp}(P_{\text{ID}})} s_f(x) =: \alpha_0$, which makes the accuracy well-defined for all relevant $\alpha \geq \alpha_0$:

$$A_{\text{OOD}}(\alpha) = \frac{P_{\text{OOD}}(X_f(\alpha), c(X) = Y)}{P_{\text{OOD}}(X_f(\alpha))}. \tag{7.2}$$

Here, Y is the correct label to the input X.

For the remainder, we consider $\alpha \in [\alpha_0, \alpha^*]$, so $A_{\text{OOD}}(\alpha) = A_{\text{ID}}(\alpha)$. Then, we have that $A_{\text{OOD}}(\alpha) = A_{\text{ID}}(\alpha) \geq \lim_{\alpha\to\infty} A_{\text{ID}}(\alpha) = A_{\text{ID}}$. The limit can be taken analogously to the proof of (b) above.

□

The first statement describes the behavior of $\mathrm{A}_{\text{OOD}}(\alpha)$ for small α and the second for large α. We observe this behavior empirically in Figure 7.3.

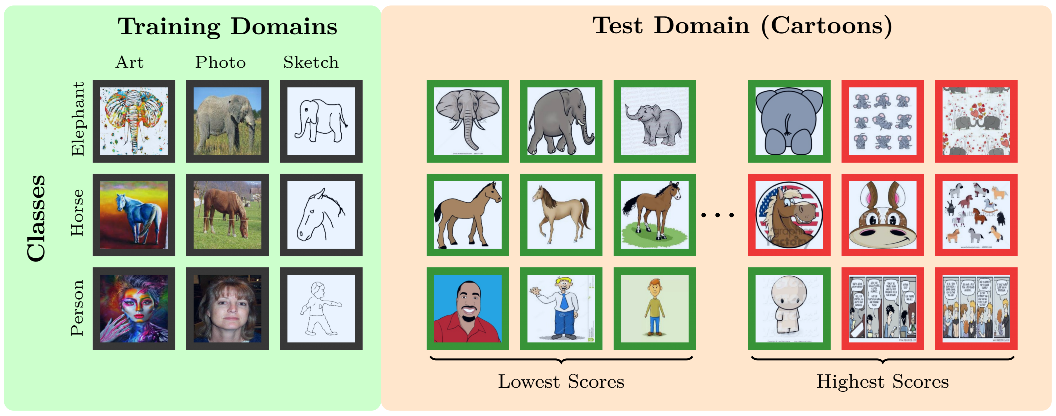


Figure 7.2. An incompetence score is able to sort out-of-distribution (OOD) images from the PACS dataset, so that higher incompetence scores result in lower classification accuracy. *(Left)* Example images from the training domains. *(Right)* Images from the test domains resulting in lowest and highest incompetence scores (using a Deep-KNN scoring function) in the feature space of a baseline ERM classifier. Green and red frames denote correctly and incorrectly classified images, respectively. Higher incompetence scores correlate with a decrease in the classifier's accuracy.

7.4. Experiments

In our experiments, we analyze the effect of an incompetence threshold α (see Subsection 7.3.2) on DG performance.[1] In the following, we first describe our experimental protocol. Then, we analyze the competence region in dependence on the threshold α and show that the competence region behaves as predicted in Proposition 8. Finally, we carry out an extensive investigation of the competence region for closed and open world settings, where we show the utility of the concept for various incompetence scores and point out current weaknesses.

As an introductory example to the competence region, we consider Figure 7.2 which depicts the experimental procedure on the PACS dataset for a standard classifier trained with Empirical Risk Minimization [ERM; 64]. We train the classifier on the domains Art, Photo, and Sketch, and apply the trained classifier in the unknown Cartoon domain. The samples in the test domain are ordered by the predicted incompetence score $s_f(x)$. As expected, the classifier still performs well on Cartoon samples with low incompetence scores (9 out of 9 classified correctly in the example), but the accuracy drops for high scores (only 2 out of 9 correct classification). Qualitatively, the score correctly notices that images with significantly different characteristics are much harder to classify. In the following sections, we quantify this behavior systematically for a number of different classifiers, incompetence scores, datasets, and domain. But first, we give details on our experimental setup.

7.4.1. Experimental Setup

We consider all combinations of nine pre-trained classifiers $c_\theta(x)$, varying both in architecture and training, nine OOD post-hoc scores $s_f(x)$ as incompetence scores on a total of 32 DG tasks from six different DG datasets. The pre-trained classifiers are obtained as follows. We train various state-of-the-art classifiers from DG literature, namely Fish [262], GroupDRO [273], SD [274], SagNet [65], Mixup [275] and VREx [126]. Furthermore, we train three different neural network architectures with empirical-risk-minimization [64]: A ResNet based architecture which we denote by ERM [276], a Vision Transformer [277] and a Swin Transformer [278]. Training details and hyperparameter settings are listed in Section C.5.

These models are trained on six domain generalization datasets from the DomainBed repository [51]: PACS [46], OfficeHome [48], VLCS [47], TerraIncognita [49], DomainNet [50] and SVIRO [279]. For an overview of these datasets see Figure 2.1. Each DG dataset consists of four to ten different domains from which we construct different DG tasks: We train a classifier on all but one domain. The one left out during training is then the OOD test domain where the competence region is evaluated. As an example consider the DG task behind the earlier example in Figure 7.2: If we train a model on the domains Photos, Art images, and Sketches, the DG task asks for an accurate model on the domain Cartoons which constitute the OOD test domain (see Figure 7.2). Overall we consider 32 DG tasks which result in 288 trained networks. We then compute the incompetence scores of each trained network In Subsection 7.3.1, we describe the process of calculating the incompetence scores.

For each DG task, we distinguish four datasets. For the ID distribution, we consider a training set, a validation set for hyperparameter optimization, and a test set that has no

[1]We provide access to our code under `https://github.com/XarwinM/competence_estimation`

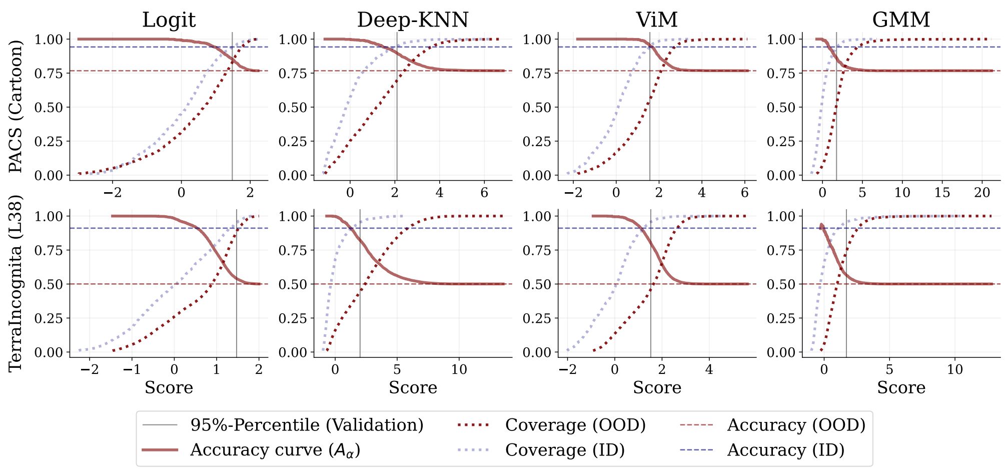


Figure 7.3. The accuracy of the ERM classifier on OOD data $\mathrm{A}_{\mathrm{OOD}}(\alpha)$ as the competence region is enlarged by increasing the incompetence threshold α. As predicted by our monotonicity criterion (Criterion 4), the accuracy starts off at $\mathrm{A}_{\mathrm{OOD}}(\alpha) \geq \mathrm{A}_{\mathrm{ID}}$ and then falls off monotonically with α. At the same time, the fraction of data the classifier is applied to increases. The classifier accuracy and fraction of considered data can easily be traded off using this figure.

influence on the optimization process for the subsequent evaluation. The classifiers are trained on the ID training set. We compute the scores for the ID distribution on the ID validation set and the ID accuracy on the ID test set. The OOD test set is given by the DG task (e.g., as in Figure 7.2). After training, we apply all post-hoc methods to the penultimate feature layer or the ouput (logits) layer of the classifier, as is typical in the OOD detection literature. If the post-hoc method needs to fit the data (as for instance with GMMs), we fit the score function on the ID training data.

7.4.2. Competence Threshold

In this section, we analyze the performance of the classifiers as a function of the threshold α which determines their competence region (see Equation 7.1 in Section 7.3). To this end, we compute the incompetence scores on the ID validation dataset and on all OOD data samples.

Figure 7.3 depicts the resulting score distributions and accuracy $\mathrm{A}_{\mathrm{OOD}}(\alpha)$ as a function of the threshold α for a single classifier (ERM). Here, we consider four incompetence scores on one of the DG tasks provided by PACS and TerraIncognita, respectively. We find that the considered incompetence scores fulfill the requirement for a competence detector in Criterion 4 that the accuracy must decrease monotonically as the threshold α increases. We then find the theoretical results in Section 7.3 confirmed: For low α, the accuracy $\mathrm{A}_{\mathrm{OOD}}(\alpha)$ is high, and even exceeds the average accuracy on the ID data A_{ID} (see Proposition 8 (a)). It eventually decreases until $\mathrm{A}_{\mathrm{OOD}}(\alpha) \to \mathrm{A}_{\mathrm{OOD}}$ for large α (see Proposition 8b).

Figure 7.3 also depicts the fraction of ID and OOD data that is considered (i.e., not rejected) as we increase the incompetence threshold α:

$$\text{Coverage}_{\text{dist}}(\alpha) = \frac{|\mathcal{D}_{\text{dist}} \cap X_f(\alpha)|}{|\mathcal{D}_{\text{dist}}|}. \tag{7.3}$$

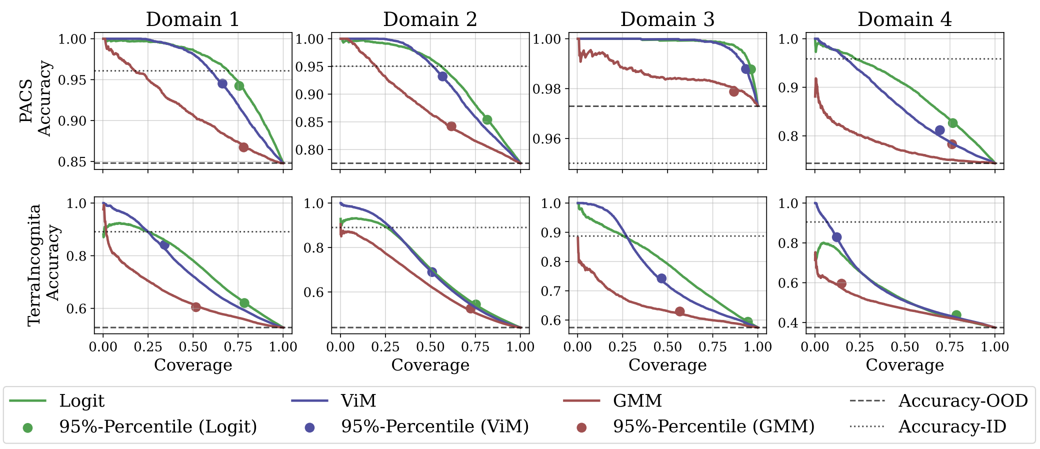


Figure 7.4. The expectation over the different domain robust classifier's accuracy on OOD data as a function of the coverage of the competence region. The 95^{th} percentiles refer to the validation set of the ID distribution and also represent averages over all classifiers (and are thus sometimes off-curve). The domains are ordered in canonical sequence as in Table C.1 in Section C.3.

When dist is equal to ID, we consider $\mathcal{D}_{\text{ID}}$ as ID validation or test set. In Figure 7.3 we opt to choose the validation set as $\mathcal{D}_{\text{ID}}$, since it is accessible during training and provides insight into what to expect during application on ID data. Note that the coverage of the ID test set is very similar to that of the ID validation set. In the case of dist being equal to OOD, we include all available OOD data. For instance, we can compare the methods at the α_{95} which includes 95% of the ID data (vertical gray line in Figure 7.3). Here, Logit keeps a significantly larger fraction of test data compared to the other incompetence scores. However, this results in a lower accuracy in the competence region $\text{A}_{\text{OOD}}(\alpha_{95})$.

Unfortunately, due to the nature of the DG problem, the accuracy curve $\text{A}_{\text{OOD}}(\alpha)$ in Figure 7.3 is not accessible during inference, which makes it difficult to choose a suitable threshold α. In Figure 7.3 we can observe that ViM and KNN achieve at the 95% percentile (with respect to the ID validation set) an accuracy that is comparable to the ID accuracy, rendering the predictions in this competence region very accurate and trustworthy. GMM and Logit obtain very high accuracies in the competence region $X_f(\alpha) \cap \mathcal{D}_{\text{OOD}}$ for small α values, but exhibit a larger drop in accuracy at the 95% percentile (w.r.t. the ID distribution). We show the accuracies $\text{A}_{\text{OOD}}(\alpha)$ for different threshold values α for all datasets and DG tasks in Section C.2.

7.4.3. Accuracy vs. Coverage Trade-Off

We illustrate the trade-off between accuracy and coverage for the ViM, GMM and Logit score for all domains in PACS and TerraIncognita in Figure 7.4. Here, we consider the empirical average over all classifiers. All scores behave relatively monotonically in the sense that increased coverage results in a reduction in accuracy.

First, it is evident that GMM (red curve) shows a non-competitive accuracy-coverage trade-off. Further, while the Logit score (green curve) exhibits a slightly favorable accuracy coverage trade-off across the PACS domain, a clear winner for TerraIncognita does not emerge. Overerall, ViM (blue curve) performs better than the Logit score in terms of accuracy in the competence region elicited via a threshold at the 95^{th} percentile of the score ID distribution. This indicates that the ViM score demonstrates greater selectivity and classifies more samples as outliers in comparison to the Logit score. However, for all cases considered, we conclude that Logit and ViM exhibit a similar accuracy coverage trade-off. Note that the curves in Figure 7.4 are not accessible when we need to set the threshold.

7.4.4. Extensive Survey

In Percentages (%)	PACS			OfficeHome			VLCS		
	OOD-Gain ↑	ID-Gain ↑	Coverage ↑	OOD-Gain ↑	ID-Gain ↑	Coverage ↑	OOD-Gain ↑	ID-Gain ↑	Coverage ↑
Deep-KNN	**11** [1-18]	**0** [-12-5]	66 [56-95]	8 [3-16]	-13 [-28-1]	82 [64-94]	**2** [0-5]	-9 [-27-14]	87 [72-99]
ViM	9 [1-19]	**0** [-17-5]	66 [50-93]	5 [2-13]	-14 [-32-1]	87 [65-95]	**2** [0-5]	**-8** [-28-14]	85 [62-99]
Softmax	7 [1-14]	-4 [-11-5]	84 [65-97]	8 [3-15]	**-12** [-32-1]	84 [67-95]	**2** [0-4]	-10 [-27-13]	93 [87-99]
Logit	9 [1-12]	-3 [-12-5]	80 [61-96]	**9** [2-16]	-13 [-33-0]	81 [66-96]	**2** [0-5]	-10 [-27-14]	92 [83-98]
Energy	9 [1-12]	-3 [-12-5]	79 [61-96]	8 [2-16]	-14 [-33-0]	82 [67-96]	**2** [0-4]	-10 [-27-14]	93 [82-98]
Energy-React	9 [1-12]	-3 [-13-5]	79 [60-96]	8 [2-16]	-14 [-33-0]	82 [67-96]	**2** [0-4]	-10 [-27-13]	93 [82-98]
Mahalonobis	1 [0-12]	-8 [-22-4]	80 [50-96]	1 [0-7]	-17 [-42-0]	91 [75-95]	0 [-1-3]	-11 [-28-14]	93 [73-99]
GMM	2 [0-13]	-8 [-21-4]	76 [50-96]	0 [0-7]	-18 [-42-0]	92 [76-95]	0 [-1-3]	-12 [-28-14]	85 [53-99]
PCA	1 [-1-10]	-12 [-21-3]	78 [57-97]	0 [0-7]	-18 [-42-0]	93 [78-96]	0 [-1-2]	-12 [-28-14]	88 [64-99]

	Terra Incognita			DomainNet			SVIRO		
	OOD-Gain ↑	ID-Gain ↑	Coverage ↑	OOD-Gain ↑	ID-Gain ↑	Coverage ↑	OOD-Gain ↑	ID-Gain ↑	Coverage ↑
Deep-KNN	**32** [12-51]	**-8** [-36-4]	37 [13-52]	**4** [0-6]	**-6** [-50-7]	85 [70-93]	**4** [1-28]	**0** [-1-0]	28 [5-64]
ViM	28 [13-51]	-13 [-35-3]	41 [7-57]	2 [0-7]	-8 [-50-6]	90 [68-97]	**4** [1-30]	**0** [0-0]	19 [6-62]
Softmax	4 [1-12]	-38 [-54–24]	85 [68-96]	3 [0-5]	-8 [-52-5]	94 [77-98]	**4** [0-24]	**0** [-10-0]	60 [28-83]
Logit	5 [1-18]	-34 [-55–23]	85 [60-98]	2 [0-5]	-8 [-51-6]	93 [77-97]	2 [0-21]	**0** [-19-0]	67 [40-87]
Energy	5 [1-19]	-33 [-55–21]	85 [55-98]	2 [0-5]	-9 [-51-5]	94 [79-98]	2 [-2-21]	**0** [-24-0]	67 [40-87]
Energy-React	5 [1-19]	-33 [-55–21]	85 [55-98]	2 [0-5]	-9 [-51-5]	93 [79-98]	2 [-2-21]	**0** [-24-0]	67 [42-88]
Mahalonobis	5 [-1-38]	-33 [-56–11]	62 [7-94]	-1 [-3-5]	-11 [-53-4]	92 [77-97]	2 [-1-28]	**0** [-19-0]	20 [5-95]
GMM	7 [-1-38]	-28 [-51–10]	56 [7-84]	-1 [-3-4]	-12 [-53-3]	93 [79-98]	3 [-1-28]	**0** [-11-0]	20 [5-67]
PCA	1 [-1-26]	-38 [-53–24]	87 [35-99]	-1 [-3-2]	-12 [-53-2]	92 [84-98]	0 [-1-15]	-2 [-26-0]	87 [47-99]

Table 7.1. Accuracy on competence region of OOD domain for different domain generalization datasets and incompetence scores. As the threshold for the competence regions, we choose the 95% percentile of the ID validation set. For all metrics, a higher value means better performance (↑). All displayed values are medians over different domain roles and classifiers, brackets indicate 90% confidence interval.

In the following, we evaluate all nine incompetence scores on all six DG datasets using the nine classifiers. Since each dataset features 32 different DG tasks, we perform a total of $32 \cdot 9 \cdot 9 = 2592$ experiments. For each experiment, we obtain accuracy curves as in Figure 7.2 as a function of α. To summarize and compare the performance of each score on each dataset, we need to deal with the trade-off between accuracy and coverage. Thus, we measure accuracy $\mathrm{A}_{\mathrm{OOD}}(\alpha_{95})$ at the score α_{95}, such that 95% of ID validation data fall below this threshold, that is $\mathrm{Frac}_{\mathrm{ID}}(\alpha_{95}) = 95\%$. As mentioned in Subsection 7.4.2, choosing α in DG is notoriously difficult, since we have no access to the test domain(s) during training. The following quantities provide useful summary statistics for comparing our results across all experiments:

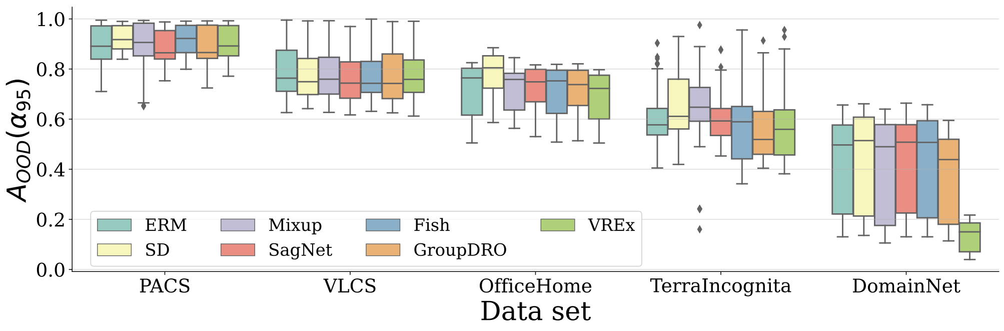


Figure 7.5. Accuracy in the competence region for different DG algorithms and OOD scores on different datasets. The chosen threshold corresponds to the 95th percentile of the ID-distribution scores. Note: VREx failed to converge with the hyperparameters used on the DomainNet dataset.

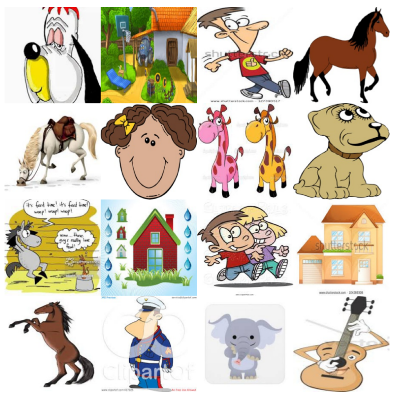
1. OOD-Gain $= \mathrm{A}_{\mathrm{OOD}}(\alpha_{95}) - \mathrm{A}_{\mathrm{OOD}}$: The performance gain in the OOD domain by considering only the data in the competence region $X_f(\alpha_{95})$.
2. ID-Gain$=$ $\mathrm{A}_{\mathrm{OOD}}(\alpha_{95}) - \mathrm{A}_{\mathrm{ID}}$: Expresses the performance gap between the accuracy on OOD data in the competence region $X_f(\alpha_{95})$ and the accuracy on the entire ID data A_{ID}.
3. Coverage $=$ $\mathrm{Coverage}_{\mathrm{OOD}}(\alpha_{95})$ as given by Equation 7.3: The proportion of OOD data that falls within the competence region.

For each quantity, a higher value indicates better performance (↑). Note that the coverage of the competence region alone is not informative. A naive approach that includes all samples in the competence region would achieve the largest competence region but would fall short in terms of OOD-Gain or ID-Gain.

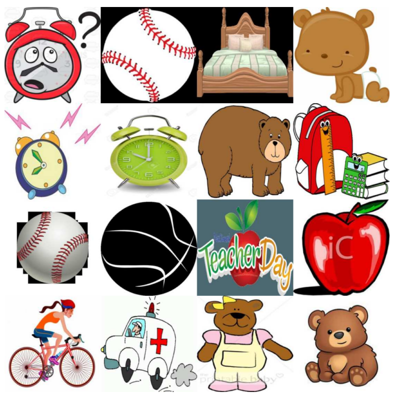
Table 7.1 summarizes the results from our extensive sweep over classifiers, datasets, and incompetence scores. The displayed values are the medians over different domain roles and classifiers. Overall, we observe that in the competence region, higher accuracy is achieved compared to the naive application on all OOD data instances. This confirms that incompetence score and accuracy are indeed tightly linked. However, for most DG datasets and incompetence scores, we are not able to replicate the ID accuracy. This indicates that we cannot naively expect the classifier to attain the same accuracy as observed in the ID distribution in the 95% percentile α_{95}. Further important findings are:

- In general, feature-based (Deep-KNN, ViM), as well as logit-based incompetence scores (Softmax, Logit, Energy, Energy-React) obtain significantly higher accuracy on OOD data (higher OOD-Gain) by filtering the data to the competence region $X_f(\alpha_{95})$ than the density- and reconstruction-based approaches (Mahalanobis, GMM, PCA).
- The feature-based scores achieve a significant performance boost on TerraIncognita. TerraIncognita contains DG tasks that suffer from a particularly huge drop in accuracy from ID to OOD distribution (see Section C.6).
- The proportion of OOD data that falls inside the competence region (i.e., coverage) is smallest for feature-based methods, but they also provide the highest accuracy across all DG datasets.

It is important to note that at the specific threshold investigated, the accuracy in the competence region remains unaffected by the DG algorithms (see Figure 7.5). Based on this observation, the incremental utility of DG algorithms specifically designed for the purpose of domain robustness becomes uncertain. If we assume that DG algorithms successfully achieve their primary goal of learning domain-invariant features, then we can speculate that they create a more valuable competence region. However, this is not in line with our observation in Figure 7.5: We observe that no domain-robust classifier is consistently and significantly more accurate in the competence region than the simple baseline classifier (ERM). In Section C.6, we also show that the same result holds without restrictions on the competence region. Furthermore, we explore in Section C.2 thresholds close to the 95th percentile and demonstrated that the relative performance of the scores remains quite consistent.



(a) Samples of domain Cartoon



(b) Open world samples for domain Cartoon.

Figure 7.6. OOD Samples (left) and open world samples (right) for the PACS Cartoon environment.

7.4.5. Open World Performance

In this section, we study how different incompetence scores shape the competence region when instances of unknown classes are present in the ID distribution. Accordingly, for each domain in PACS, VLCS, Office-Home, and TerraIncognita datasets, we create a matching "open world“ domain containing only instances of unknown classes. In total, we create 16 open world domains. For example, if we evaluate a model on the PACS Cartoon domain, we create an open world domain containing only cartoons of classes that are not in the PACS dataset as demonstrated in Figure 7.6. We describe the procedure for creating the open world domains in detail in Section C.4. In the following, we restrict our analysis to the 16 domains for which an open world twin exists.

We enrich the existing test domains with 0%, 5%, 10%, 15%, 20%, and 25% instances with unknown classes. A good incompetence score should mark instances of unknown classes with a high value and therefore render them outside of the competence region $X_c(\alpha_{95})$. In this case, the OOD-Gain would increase as more open world instances find their way into the test set. In Figure 7.7 we observe that this behavior is achieved particularly well for the ViM score. The Logit and Softmax scores are less successful in delineating unknown class instances from the competence region and therefore the OOD-Gain is less pronounced.

Indeed, to test whether this observation holds statistically across all classifiers, we fit a hierarchical linear regression [280] on OOD Gain with `Classifier`, `Percentage`

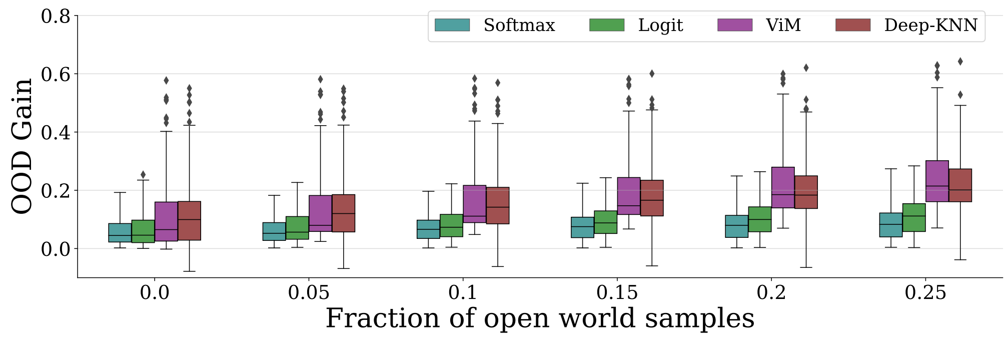


Figure 7.7. Performance of Logit and Softmax scores (logit-based) against Deep-KNN and ViM (feature-based) for an increasing fraction of open world data (unknown classes) in the test domain. The performance gain on the OOD data (*OOD-Gain*, higher is better) for the logit-based methods is less pronounced compared to ViM and Deep-KNN.

`Open World`, and `Incompetence Score`, as well as their interactions as fixed factors, together with `Data Set` and `Test Domain` as random factors (to account for the fact that the same classifier is evaluated in multiple data sets and test domains). The statistical results confirm the general trends visible in Figure 7.7. First, we find significant main effects of `Percentage Open World` (i.e., overall OOD Gain increases with an increasing number of open world instances) and `Incompetence Score` (i.e., ViM and Deep-KNN achieve a higher overall OOD Gain). Importantly, the only significant interaction revealed by the hierarchical regression model suggests that ViM is able to achieve the largest OOD Gain as the fraction of open world samples increases. Note, that the same trend is present for Deep-KNN, but it fails to achieve statistical significance due to its high variability (see Figure 7.7). Moreover, none of the effects involving the factor `Classifier` turn out to be significant predictors of OOD Gain, suggesting that the results are largely classifier-independent.

In the closed world setting, differences between logit- and feature-based scores are for most DG data sets small (see e.g. Table 7.1). However, we have shown that it is very relevant in the setting where instances of unknown classes occur. In Section C.4 we show the open world behavior for all incompetence scores.

7.4.6. Estimating the Incompetence Threshold

Choosing the 95% percentile of the ID distribution as incompetence threshold can be considered as weighting the trade-off between accuracy and coverage towards coverage – only 5% of ID data are rejected. We now seek a slightly different incompetence threshold which puts more weight on the accuracy. The question we want to address is whether *we can set a threshold such that a certain accuracy is achieved in the competence region?* This question is of high practical relevance, but also particularly challenging for two reasons. First, many scores used so far have no out-of-the-box connection to the accuracy and second, we deal with a domain shift that might result in a completely new score-accuracy relationship.

Thus, as a potential remedy, we suggest learning $\widetilde{s}_f(x) = p_{\mathrm{ID}}(f(x) \neq y \mid s_f(x))$ and using this conditional probability as a *transformed* score. This score represents the probability of an incorrect prediction given the original score. If we define a competence region with an incompetence threshold of $1 - \mathrm{A}_{\mathrm{ID}}$, we can expect an accuracy of at least A_{ID} on ID data. We hope that this relation also holds under domain shift. To predict $\widetilde{s}_f(x) =$

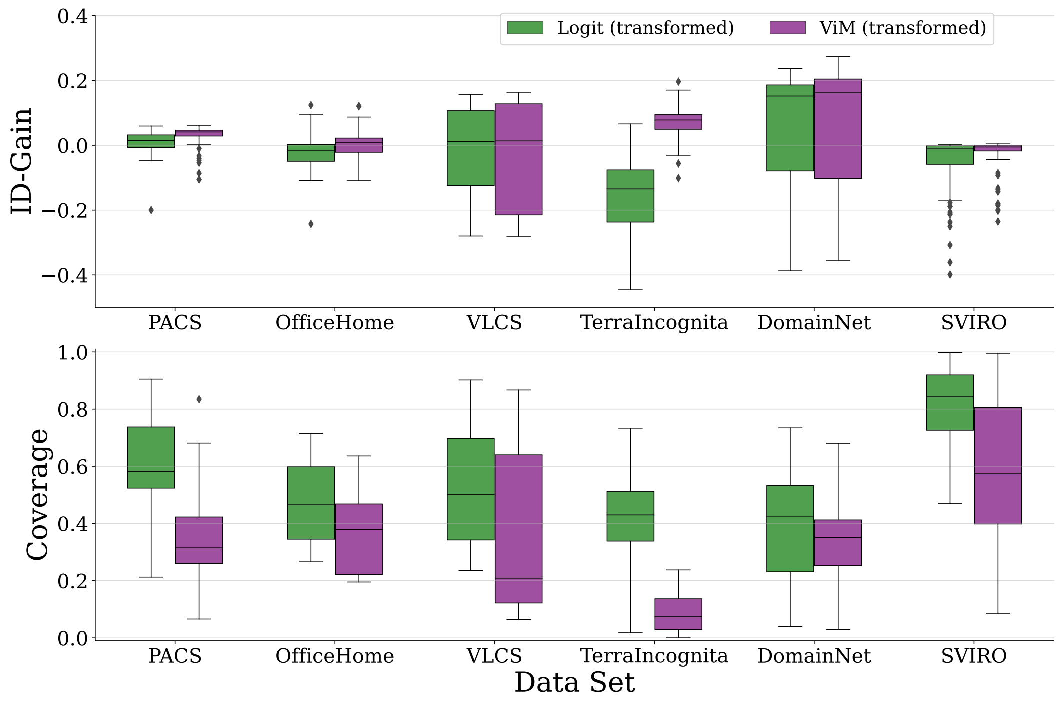


Figure 7.8. ID-Gain and Coverage for Logit and ViM if transformed as described in Subsection 7.4.6. The threshold is set such that the ID-Gain should be at least 0. *Top row:* ID-Gain for Logit and ViM due to different datasets. *Bottom row:* Coverage for Logit and ViM across different DG datasets. Medians and quantiles for all boxplots are computed over different domain roles and classifiers. The threshold is set as the ID accuracy.

$p_{\mathrm{ID}}(f(x) \neq y \mid s_f(x))$ we rely on an architecture that is constrained to be monotonic as proposed in [31]. Therefore, we do not change the order of the scores and equip the transformed score with an inductive bias that is consistent with Criterion 4. The transformed score also has a predictable extrapolation behavior which is helpful when the distribution shifts. Note that since the transformation is monotonic, a threshold for the original score is also a valid threshold for the transformed score and *vice versa*. Therefore, we can also interpret this approach as estimating an incompetence threshold such that a certain accuracy is achieved.

Accordingly, Figure 7.8 depicts the ID-Gain and Coverage for ViM and Logit (transformed), if we select $1 - \mathrm{A}_{\mathrm{ID}}$ as the incompetence threshold. The transformed ViM score suggests that we achieve in most cases at least the ID accuracy, but at the cost of small coverage. The transformed Logit score has higher coverage, but it often fails to reproduce the ID accuracy (e.g., in the TerraIncognita data set). However, while we attain the ID accuracy for most cases, we still observe some failure cases, which makes the approach only tentative. Note, that these results also suggest that the information contained in the logits is not sufficient to give suitable competence regions in the sense of our question.

7.5. Conclusion

Accepting only predictions from the competence region of a classifier increases its accuracy dramatically under domain shift. Determining the fraction of samples where the classifier could be considered competent is a question of how to approach the trade-off between accuracy and coverage. Addressing this trade-off via the incompetence threshold is application-dependent and particularly challenging in the domain generalization (DG) setting where the test distribution differs from the training distribution per definition. Still, we showed that even in DG, it is possible to achieve higher than in-distribution accuracy under domain shift – at the price of potentially diminished coverage (see Figure 7.2 or Section C.2).

Furthermore, we investigated a coverage-oriented threshold that would reject only a pre-defined fraction (e.g., 5%) of all instances from the training distribution. In this case, we achieved a considerable improvement under distribution shift compared to a naive application where no samples are rejected. However, at this particular threshold, we could recover the ID accuracy only in some settings. Thus, we also studied whether we can learn an accuracy-oriented threshold where some predefined ID accuracy is guaranteed in the competence region. This approach was able to replicate the ID accuracy in the competence region for most investigated domain shifts. However, for a few domains, OOD accuracy drops significantly below the expected ID, calling for a more detailed understanding of the behavior of incompetence scores in DG. Nevertheless, we observed that accuracy in the competence region behaves monotonically with the threshold α (see Proposition 8 and Subsection 7.4.2).

Finally, we investigated differences between the closed and open world settings. We found that in the open world setting, feature-based methods, such as Deep-KNN [244] and ViM [245], elicit a particularly useful competence region. In a closed world DG setting, a clear winner does not emerge, but ViM and Deep-KNN seem to be competitive to logit-based approaches. We also analyzed whether we could find differences in the accuracy of the competence region with respect to different classifiers. We could not find statistically significant effects on the accuracy in the competence region, leaving the benefit of robust algorithms for DG and different architectures for enlarged competence regions questionable.

All post-hoc methods investigated in this chapter are comparably fast to evaluate and therefore easily accessible for practitioners. However, the resolution of the trade-off between accuracy and coverage is not yet satisfactory in all cases, calling for more research on better competence scores. One interesting avenue concerns the use of *multivariate* scores (i.e., a combination of multiple scores) with the potential to elicit better competence regions.

8 Summary and Conclusions

Learning accurate and robust models is in general only feasible if we take the kind of distribution shift into account. Understanding the need to consider the distribution shift leads us in three key directions. First, we characterize different distribution shifts and relate them to different types of invariances. Second, we derive two frameworks that deal with distinct distribution shifts, one based on the principle of ICM – a fundamental concept in causality – and the other based on contextual knowledge of the environments from which the data originates. Third, appreciating the challenges that robustness poses, we explore the concept of a competence region to identify samples where the predictive model can be considered incompetent or competent to enhance its trustworthiness.

8.1. Our Work

Invariances and Distribution Shifts In this work, we recap basic concepts of causality and deep learning and relate them to robustness when possible. Most importantly, we explored the effect of different types of invariances on robustness under various distribution shifts. We could therefore systematically answer the question of which forms of invariances promise robustness under which kind of distribution shift (see Section 3.8). An interesting future research direction is the identification and characterization of the type of distribution shift in Domain Generalization (DG) datasets. This poses a hard challenge, particularly in cases where the relevant variables are latent variables.

Using the Principle of ICM for Robustness Our work in Chapter 5 considers a specific form of invariance that we term *causal invariance*. This type of invariance is rooted in the principle of ICM which is a fundamental concept in causality (see Section 3.6). We have operationalized this principle into an objective amenable to gradient-based optimization. Therefore, our approach eliminates scalability issues seen in combinatorial optimization. Furthermore, our use of normalizing flows in an information-theoretic context extends the additive noise model, allowing the identification of relevant variables, even when the presented variables themselves lack inherent meaning, such as pixels in image data. Additionally, we proved theoretically that models trained in our framework identify the true underlying causal relations under suitable conditions. We further demonstrated empirically that we are able to identify the invariances that promise robustness in new environments. Employing a gating architecture optimized via gradient-descent, we successfully excluded non-causally relevant variables, enabling the identification and interpretation of the true causes of a target variable. Besides the potential future adaptations discussed in Section 5.6, we think that the framework could be extended to deal with the invariances discussed in Section 2.4. An intriguing approach would be to train a multi-invariance model capable of choosing between different invariances (e.g., in the form of a condition).

Context-Aware DG We proposed a novel approach that utilizes context information about the data's origin. This enables our model to adapt its predictions to the characteristics of the environments. We establish crucial criteria that are necessary for our approach to yield benefits. In addition to proving their necessity, we also demonstrate empirically that two of these criteria are readily accessible with standard models, enabling the verification of potential advantages offered by our approach. Additionally, we characterize the kind of distribution shift necessary for our approach to deliver advantages. We empirically showcase the benefits of our approach over baseline models in several scenarios.

Although our approach shows the benefits of contextual information, it can also encounter failure cases. We demonstrate that distribution shifts can be detected, identifying potential failure cases. The detection of novel environments extends the possibilities of our approach to model selection. Specifically, we show how we can select between the most predictive model (regarding ID data) and the most robust model to overcome the inherent trade-off between robustness and predictive in the ID setting. While we opened a new avenue in Domain Generalization by exploiting context information, there remain several interesting research directions ready to be explored.

While we explored the concept within the supervised learning paradigm, our approach holds promise in various other domains. It could be adapted for domain disentanglement, data generation tasks, or seamlessly integrated with alternative DG methodologies. Another intriguing avenue involves inferring environments (e.g., through k-nearest neighbor search in feature space) and exploiting them within our framework. Additionally, investigating the impact of regularization and inductive biases in our approach presents another compelling research direction. In our work in Chapter 7, we delved into the inherent trade-off between coverage and accuracy. Exploring similar trade-offs in the context of novel environment detection would be an exciting avenue for further research. And finally, throughout our investigation in Chapter 6, we showed the benefits of context-aware networks mainly on synthetical or half-synthetical datasets. A large-scale investigation of our approach on several real-world datasets is therefore desirable.

Competence Region In our work in Chapter 7, we explored various post-hoc OOD detection methods to define the competence region. We demonstrated significant performance gains by rejecting samples outside a classifier's competence region, even under distribution shift. Choosing different thresholds for where the classifier can be deemed competent leads to the fundamental trade-off between accuracy and coverage. By manipulating this threshold, we demonstrated that the accuracy behaves monotonically with the threshold: The more competent the classifier, the higher the accuracy, but at the price of little coverage (and vice versa for high coverage). Notably, we achieved above ID accuracy levels for OOD samples in the region of high competence across several datasets. In the open-world scenario where new classes can occur in the unknown test environment, we observed that feature-based OOD detection methods outperformed those based on logits and the softmax output. Additionally, we found that the competence regions of standard classifiers are comparable to that of a set of DG methods.

Although, we considerably improved the accuracy on OOD data within the competence region, we encountered difficulties in determining an adequate threshold that ensures the classifier's ID accuracy. This highlights the need for future research to implement the competence region more effectively. Improving the competence region could involve exploiting deep neural networks on multiple scales and combining various OOD scores. This might also provide insights into the reasons for the model's silent failure. Exploring the competence region in regression tasks presents another interesting avenue for future research. It is important to note that our approach is exceptionally user-friendly since the post-hoc methods leave the classifier unchanged and are easily applicable.

No matter how robust and accurate a method is, in a sufficiently complex scenario, failure cases might occur. Therefore, the concept of a competence region remains always important – specifically, if we are dealing with safety-critical situations. Considering the possibility of a reject option should therefore be standard in all safety-critical applications.

8.2. The Broader Perspective

From a broader perspective, the most effective methods on many relevant robustness tasks employ feature extractors learned on extensive datasets, which tend to be very generalizable (see Subsection 2.2.3). However, this approach is not universally applicable. First, if there is not enough data in the problem domain to train a large-scale feature extractor due to data scarcity or data protection laws, a large-scale feature extractor is not achievable. Second, certain distribution shifts exhibit problems that do not stem from model fitting, but rather from identifying the right invariances. In this case, more data does not necessarily help (see for instance Subsection 5.5.2). Lastly, large-scale models might still suffer from silent failures emphasizing the role of competence regions or similar concepts. Nonetheless, large-scale feature extractors trained on extensive datasets offer intriguing research opportunities. For instance, these feature extractors could also be employed in our and similar frameworks: instead of the data space, we could apply our methods in the feature space. This is possible with a context-aware neural network (see Chapter 6), with our invariant learning (see Chapter 6) and within the setting of competence regions (see Chapter 7). The impact of large-scale feature extractors on the realms of robustness and DG will unfold in the coming years and will answer the question of whether DG algorithms seamlessly integrate large-scale feature extractors or hold a specialized role in limited data scenarios and niche problems.

In classical supervised learning, as well as in most other machine learning paradigms, a single sample within a dataset indicates an algorithm's performance. As the test set comprises numerous samples, we can statistically estimate the algorithm's success on the task. When it comes to distribution shifts, model evaluation faces an added layer of complexity: the assessment of performance in novel environments. This assessment presents a challenge due to the scarce amount of available environments and finite data therein, resulting in a very noisy performance evaluation. Another way to put it: in supervised learning, each sample provides a single signal for an algorithm's success, while in robustness, domain generalization, and causality, each distribution shift[1] offers a noisy individual signal for an algorithm's success (see also Subsection 2.2.1). As a result, evaluating robust algorithms and causal discovery algorithms is more challenging. Successful outcomes must be approached cautiously since they are often tested only on a few distribution shifts (*environment-scarcity*).

[1] In causal discovery, each dataset can be considered as one sample from an evaluation standpoint

Furthermore, the identification and characterization of distribution shifts in practical applications is unfortunately not established. However, this kind of understanding is of crucial importance for choosing the type of invariance to seek as shown in Section 3.8. Moreover, large-scale trained feature extractors might excel for certain distribution shifts, like *covariate shifts*, but might fail for other dataset shifts such as the *source component shift*. Therefore, understanding the kind of distribution shift might be indispensable in many applications. In Chapter 6 we took first steps in this direction by characterizing the distribution shift where our approach might be beneficial and by formulating criteria that are testable and necessary for our method to yield benefits. It is important to mention that the absence of considering the distribution shift at hand could potentially explain why many DG algorithms do not perform better than a naive baseline on many benchmark datasets as discussed in Section 3.8.

The fields of robustness and DG stand as crucial areas of research. Despite the substantial efforts in these fields, there remains a lack of foundational comprehension and effective evaluation processes. The recent appearance of various benchmark datasets facilitates method comparisons, yet a nuanced understanding of the specific distribution shifts within the datasets is essential to ensure fair model comparison, as we have elaborated in Section 3.8. In the coming years, one of the big challenges for DG is the development of well-founded evaluation processes, including the consideration of the distribution shift at hand. Progress in this direction promises not only to enhance algorithmic development but also to provide a fundamental understanding of their current state.

Bibliography

[1] P. Domingos, “A few useful things to know about machine learning,” *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, 2012.

[2] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of machine learning*. MIT press, 2018.

[3] T. Kriete, D. C. Noelle, J. D. Cohen, and R. C. O’Reilly, “Indirection and symbol-like processing in the prefrontal cortex and basal ganglia,” *Proceedings of the National Academy of Sciences*, vol. 110, no. 41, pp. 16390–16395, 2013.

[4] T. R. Besold and U. Schmid, “Why generality is key to human-level artificial intelligence,” *Advances in Cognitive Systems*, vol. 4, pp. 13–24, 2016.

[5] R. Geirhos, C. R. Temme, J. Rauber, H. H. Schütt, M. Bethge, and F. A. Wichmann, “Generalisation in humans and deep neural networks,” *Advances in neural information processing systems*, vol. 31, 2018.

[6] Z. Shen, J. Liu, Y. He, X. Zhang, R. Xu, H. Yu, and P. Cui, “Towards out-of-distribution generalization: A survey,” *arXiv:2108.13624*, 2021.

[7] N. Meinshausen, “Causality from a distributional robustness point of view,” in *2018 IEEE Data Science Workshop (DSW)*, pp. 6–10, IEEE, 2018.

[8] J. Quinonero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence, *Dataset shift in machine learning*. Mit Press, 2008.

[9] P. Bandi, O. Geessink, Q. Manson, M. Van Dijk, M. Balkenhol, M. Hermsen, B. E. Bejnordi, B. Lee, K. Paeng, A. Zhong, *et al.*, “From detection of individual metastases to classification of lymph node status at the patient level: the camelyon17 challenge,” *IEEE transactions on medical imaging*, vol. 38, no. 2, pp. 550–560, 2018.

[10] G. Christie, N. Fendley, J. Wilson, and R. Mukherjee, “Functional map of the world,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6172–6180, 2018.

[11] R. Geirhos, K. Narayanappa, B. Mitzkus, T. Thieringer, M. Bethge, F. A. Wichmann, and W. Brendel, “Partial success in closing the gap between human and machine vision,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 23885–23899, 2021.

[12] D. Hendrycks and T. Dietterich, “Benchmarking neural network robustness to common corruptions and perturbations,” *arXiv preprint arXiv:1903.12261*, 2019.

[13] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, “Do imagenet classifiers generalize to imagenet?,” in *International conference on machine learning*, pp. 5389–5400, PMLR, 2019.

[14] P. W. Koh, S. Sagawa, H. Marklund, S. M. Xie, M. Zhang, A. Balsubramani, W. Hu, M. Yasunaga, R. L. Phillips, I. Gao, *et al.*, "Wilds: A benchmark of in-the-wild distribution shifts," in *International Conference on Machine Learning*, pp. 5637–5664, PMLR, 2021.

[15] K. Muandet, D. Balduzzi, and B. Schölkopf, "Domain generalization via invariant feature representation," in *International conference on machine learning*, pp. 10–18, PMLR, 2013.

[16] J. Wang, C. Lan, C. Liu, Y. Ouyang, T. Qin, W. Lu, Y. Chen, W. Zeng, and P. Yu, "Generalizing to unseen domains: A survey on domain generalization," *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[17] K. Zhou, Z. Liu, Y. Qiao, T. Xiang, and C. C. Loy, "Domain generalization: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

[18] F. Rosenblatt, *The perceptron, a perceiving and recognizing automaton Project Para.* Cornell Aeronautical Laboratory, 1957.

[19] T. J. Sejnowski, *The deep learning revolution.* MIT press, 2018.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[21] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[22] A. H. Marblestone, G. Wayne, and K. P. Kording, "Toward an integration of deep learning and neuroscience," *Frontiers in computational neuroscience*, vol. 10, p. 94, 2016.

[23] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Žídek, A. W. Nelson, A. Bridgland, *et al.*, "Improved protein structure prediction using potentials from deep learning," *Nature*, vol. 577, no. 7792, pp. 706–710, 2020.

[24] D. Guest, K. Cranmer, and D. Whiteson, "Deep learning and its application to lhc physics," *Annual Review of Nuclear and Particle Science*, vol. 68, pp. 161–181, 2018.

[25] S. J. Russell and P. Norvig, *Artificial intelligence a modern approach.* London, 2010.

[26] D. Milmo and agency, "Chatgpt reaches 100 million users two months after launch." `https://www.theguardian.com/technology/2023/feb/02/chatgpt-100-million-users-open-ai-fastest-growing-app`.

[27] Y. LeCun and L. Friedman, "Yann lecun: Dark matter of intelligence and self-supervised learning | lex fridman podcast #258." `https://www.youtube.com/watch?v=SGzMElJ11Cc&ab_channel=LexFridman` at time 1:51:35.

[28] B. Marr, "The amazing ways google uses deep learning ai." `https://www.forbes.com/sites/bernardmarr/2017/08/08/the-amazing-ways-how-google-uses-deep-learning-ai/?sh=2cffb69f3204`.

[29] J. Peters, P. Bühlmann, and N. Meinshausen, "Causal inference by using invariant prediction: identification and confidence intervals," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 78, no. 5, pp. 947–1012, 2016.

[30] P. Bühlmann, “Invariance, causality and robustness,” 2020.

[31] J. Müller, R. Schmier, L. Ardizzone, C. Rother, and U. Köthe, “Learning robust models using the principle of independent causal mechanisms,” in *DAGM German Conference on Pattern Recognition*, pp. 79–110, Springer, 2021.

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

[33] S. Magliacane, T. van Ommen, T. Claassen, S. Bongers, P. Versteeg, and J. M. Mooij, “Domain adaptation by using causal inference to predict invariant conditional distributions,” in *Advances in Neural Information Processing Systems*, pp. 10846–10856, 2018.

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

[35] Y. Geifman and R. El-Yaniv, “Selective classification for deep neural networks,” *Advances in neural information processing systems*, vol. 30, 2017.

[36] R. El-Yaniv *et al.*, “On the foundations of noise-free selective classification.,” *Journal of Machine Learning Research*, vol. 11, no. 5, 2010.

[37] J. Yang, K. Zhou, Y. Li, and Z. Liu, “Generalized out-of-distribution detection: A survey,” *arXiv:2110.11334*, 2021.

[38] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[39] J. Müller, L. Ardizzone, and U. Köthe, “Prodas: Probabilistic dataset of abstract shapes,” *Heidelberg University Library doi:10.11588/HEIDOK.00034135*, 2023.

[40] J. Müller, L. Kühmichel, M. Rohbeck, S. T. Radev, and U. Köthe, “Towards context-aware domain generalization: Representing environments with permutation-invariant networks,” *arXiv preprint arXiv:2312.10107*, 2023.

[41] J. Müller, S. T. Radev, R. Schmier, F. Draxler, C. Rother, and U. Köthe, “Finding competence regions in domain generalization,” *Transactions on Machine Learning Research*, 2023.

[42] H. Ye, C. Xie, T. Cai, R. Li, Z. Li, and L. Wang, “Towards a theoretical framework of out-of-distribution generalization,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 23519–23531, 2021.

[43] D. Mahajan, S. Tople, and A. Sharma, “Domain generalization using causal matching,” in *International Conference on Machine Learning*, pp. 7313–7324, PMLR, 2021.

[44] I. J. Goodfellow, J. Shlens, and C. Szegedy, “Explaining and harnessing adversarial examples,” *arXiv:1412.6572*, 2014.

[45] Y. Li, Y. Yang, W. Zhou, and T. Hospedales, “Feature-critic networks for heterogeneous domain generalization,” in *International Conference on Machine Learning*, pp. 3915–3924, PMLR, 2019.

[46] D. Li, Y. Yang, Y.-Z. Song, and T. M. Hospedales, “Deeper, broader and artier domain generalization,” in *Proceedings of the IEEE international conference on computer vision*, pp. 5542–5550, 2017.

[47] C. Fang, Y. Xu, and D. N. Rockmore, “Unbiased metric learning: On the utilization of multiple datasets and web images for softening bias,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1657–1664, 2013.

[48] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan, "Deep hashing network for unsupervised domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5018–5027, 2017.

[49] S. Beery, G. Van Horn, and P. Perona, "Recognition in terra incognita," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 456–473, 2018.

[50] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang, "Moment matching for multi-source domain adaptation," in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1406–1415, 2019.

[51] I. Gulrajani and D. Lopez-Paz, "In search of lost domain generalization," *arXiv preprint arXiv:2007.01434*, 2020.

[52] M. Wang and W. Deng, "Deep visual domain adaptation: A survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[53] C. Xu, D. Tao, and C. Xu, "A survey on multi-view learning," *arXiv preprint arXiv:1304.5634*, 2013.

[54] R. Volpi, H. Namkoong, O. Sener, J. C. Duchi, V. Murino, and S. Savarese, "Generalizing to unseen domains via adversarial data augmentation," *Advances in neural information processing systems*, vol. 31, 2018.

[55] X. Yue, Y. Zhang, S. Zhao, A. Sangiovanni-Vincentelli, K. Keutzer, and B. Gong, "Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2100–2110, 2019.

[56] F. Qiao, L. Zhao, and X. Peng, "Learning to learn single domain generalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12556–12565, 2020.

[57] A. H. Liu, Y.-C. Liu, Y.-Y. Yeh, and Y.-C. F. Wang, "A unified feature disentangler for multi-domain image translation and manipulation," *Advances in neural information processing systems*, vol. 31, 2018.

[58] F. M. Carlucci, A. D'Innocente, S. Bucci, B. Caputo, and T. Tommasi, "Domain generalization by solving jigsaw puzzles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2229–2238, 2019.

[59] D. Kim, Y. Yoo, S. Park, J. Kim, and J. Lee, "Selfreg: Self-supervised contrastive regularization for domain generalization," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9619–9628, 2021.

[60] M. Ilse, J. M. Tomczak, C. Louizos, and M. Welling, "Diva: Domain invariant variational autoencoders," in *Medical Imaging with Deep Learning*, pp. 322–348, PMLR, 2020.

[61] H. Zhang, N. Dullerud, L. Seyyed-Kalantari, Q. Morris, S. Joshi, and M. Ghassemi, "An empirical framework for domain generalization in clinical settings," in *Proceedings of the conference on health, inference, and learning*, pp. 279–290, 2021.

[62] F. Pfisterer, C. Harbron, G. Jansen, and T. Xu, "Evaluating domain generalization for survival analysis in clinical studies," in *Conference on Health, Inference, and Learning*, pp. 32–47, PMLR, 2022.

[63] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, pp. 8748–8763, PMLR, 2021.

[64] V. N. Vapnik, "An overview of statistical learning theory," *IEEE transactions on neural networks*, vol. 10, no. 5, pp. 988–999, 1999.

[65] H. Nam, H. Lee, J. Park, W. Yoon, and D. Yoo, "Reducing domain gap by reducing style bias," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8690–8699, 2021.

[66] B. Sun and K. Saenko, "Deep coral: Correlation alignment for deep domain adaptation," in *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, pp. 443–450, Springer, 2016.

[67] D. Li, Y. Yang, Y.-Z. Song, and T. Hospedales, "Learning to generalize: Meta-learning for domain generalization," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, 2018.

[68] A. Ng and M. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," *Advances in neural information processing systems*, vol. 14, 2001.

[69] E. W. Weisstein, "Invariant.." `https://mathworld.wolfram.com/Invariant.html`.

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

[71] A. T. Nguyen, T. Tran, Y. Gal, and A. G. Baydin, "Domain invariant representation learning with domain density transformations," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5264–5275, 2021.

[72] M. Rojas-Carulla, B. Schölkopf, R. Turner, and J. Peters, "Invariant models for causal transfer learning," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 1309–1342, 2018.

[73] N. Pfister, P. Bühlmann, and J. Peters, "Invariant causal prediction for sequential data," *Journal of the American Statistical Association*, vol. 114, no. 527, pp. 1264–1276, 2019.

[74] R. Tachet des Combes, H. Zhao, Y.-X. Wang, and G. J. Gordon, "Domain adaptation with conditional distribution matching and generalized label shift," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19276–19289, 2020.

[75] M. Gong, K. Zhang, T. Liu, D. Tao, C. Glymour, and B. Schölkopf, "Domain adaptation with conditional transferable components," in *International conference on machine learning*, pp. 2839–2848, PMLR, 2016.

[76] K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang, "Domain adaptation under target and conditional shift," in *International conference on machine learning*, pp. 819–827, PMLR, 2013.

[77] Z. Lipton, Y.-X. Wang, and A. Smola, "Detecting and correcting for label shift with black box predictors," in *International conference on machine learning*, pp. 3122–3130, PMLR, 2018.

[78] M. C. Du Plessis and M. Sugiyama, "Semi-supervised learning of class balance under class-prior change by distribution matching," *Neural Networks*, vol. 50, pp. 110–119, 2014.

[79] P. Stojanov, Z. Li, M. Gong, R. Cai, J. Carbonell, and K. Zhang, "Domain adaptation with invariant representation learning: What transformations to learn?," *Advances in Neural Information Processing Systems*, vol. 34, pp. 24791–24803, 2021.

[80] K. Zhang, V. Zheng, Q. Wang, J. Kwok, Q. Yang, and I. Marsic, "Covariate shift in hilbert space: A solution via sorrogate kernels," in *International Conference on Machine Learning*, pp. 388–395, PMLR, 2013.

[81] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[82] M. Long, Y. Cao, J. Wang, and M. Jordan, "Learning transferable features with deep adaptation networks," in *International conference on machine learning*, pp. 97–105, PMLR, 2015.

[83] N. Courty, R. Flamary, A. Habrard, and A. Rakotomamonjy, "Joint distribution optimal transportation for domain adaptation," *Advances in neural information processing systems*, vol. 30, 2017.

[84] J. Shen, Y. Qu, W. Zhang, and Y. Yu, "Wasserstein distance guided representation learning for domain adaptation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[85] T. Vigen, "Spurious correlations." `https://www.tylervigen.com/spurious-correlations`.

[86] F. H. Messerli, "Chocolate consumption, cognitive function, and nobel laureates," *N Engl J Med*, vol. 367, no. 16, pp. 1562–1564, 2012.

[87] J. Mooij and T. Heskes, "Cyclic causal discovery from continuous equilibrium data," *arXiv preprint arXiv:1309.6849*, 2013.

[88] H. Reichenbach, *The direction of time*, vol. 65. Univ of California Press, 1956.

[89] B. Schölkopf, "Causality for machine learning," in *Probabilistic and Causal Inference: The Works of Judea Pearl*, pp. 765–804, 2022.

[90] J. Pearl, "A constraint-propagation approach to probabilistic reasoning," in *Proceedings of the First Conference on Uncertainty in Artificial Intelligence*, pp. 31–42, 1985.

[91] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann, 1988.

[92] A. P. Dawid, "Statistical causality from a decision-theoretic perspective," *Annual Review of Statistics and Its Application*, vol. 2, pp. 273–303, 2015.

[93] J. M. Mooij, S. Magliacane, and T. Claassen, "Joint causal inference from multiple contexts," *arXiv preprint arXiv:1611.10351*, 2016.

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

[95] K. R. Popper, *The logic of scientific discovery*. Basic Books, 1959.

[96] D. Buchsbaum, S. Bridgers, D. Skolnick Weisberg, and A. Gopnik, "The power of possibility: Causal learning, counterfactual reasoning, and pretend play," *Philosophical Transactions of the Royal Society B: Biological Sciences*, vol. 367, no. 1599, pp. 2202–2212, 2012.

[97] B. Schölkopf and J. von Kügelgen, "From statistical to causal learning," *arXiv preprint arXiv:2204.00607*, 2022.

[98] S. L. Lauritzen, *Graphical Models*. Oxford University Press, 1996.

[99] J. Pearl and D. Mackenzie, *The book of why: the new science of cause and effect*. Basic Books, 2018.

[100] J. Ellenberg, *How not to be wrong: The power of mathematical thinking*. Penguin, 2015.

[101] J. Berkson, "Limitations of the application of fourfold table analysis to hospital data," *Biometrics Bulletin*, vol. 2, no. 3, pp. 47–53, 1946.

[102] T. Verma and J. Pearl, "Equivalence and synthesis of causal models," in *Proceedings of the Sixth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 255–270, 1990.

[103] P. Spirtes, C. N. Glymour, and R. Scheines, *Causation, prediction, and search*. MIT press, 2000.

[104] J. NEYMAN, "On the application of probability theory to agricultural experiments: essay on principles, section 9," *Statistical Science*, vol. 5, pp. 465–480, 1923.

[105] R. A. Fisher, *Statistical Methods for Research Workers*. No. 3, Oliver and Boyd, 1925.

[106] D. B. Rubin, "Estimating causal effects of treatments in randomized and nonrandomized studies.," *Journal of educational Psychology*, vol. 66, no. 5, p. 688, 1974.

[107] D. B. Rubin, "Causal inference using potential outcomes: Design, modeling, decisions," *Journal of the American Statistical Association*, vol. 100, no. 469, pp. 322–331, 2005.

[108] S. L. Morgan and C. Winship, *Counterfactuals and causal inference*. Cambridge University Press, 2015.

[109] G. W. Imbens and D. B. Rubin, *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press, 2015.

[110] P. W. Holland, "Statistics and causal inference," *Journal of the American statistical Association*, vol. 81, no. 396, pp. 945–960, 1986.

[111] J. Pearl, "Bayesian networks: A model of self-activated memory for evidential reasoning," in *Proceedings of the 7th conference of the Cognitive Science Society, University of California, Irvine, CA, USA*, pp. 15–17, 1985.

[112] P. Daniusis, D. Janzing, J. Mooij, J. Zscheischler, B. Steudel, K. Zhang, and B. Schölkopf, "Inferring deterministic causal relations," *arXiv preprint arXiv:1203.3475*, 2012.

[113] N. Shajarisales, D. Janzing, B. Schölkopf, and M. Besserve, "Telling cause from effect in deterministic linear dynamical systems," in *International Conference on Machine Learning*, pp. 285–294, PMLR, 2015.

[114] D. Janzing and B. Schölkopf, "Causal inference using the algorithmic markov condition," *IEEE Transactions on Information Theory*, vol. 56, no. 10, pp. 5168–5194, 2010.

[115] B. Schölkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. Mooij, “On causal and anticausal learning,” *arXiv preprint arXiv:1206.6471*, 2012.

[116] H. Shimodaira, “Improving predictive inference under covariate shift by weighting the log-likelihood function,” *Journal of statistical planning and inference*, vol. 90, no. 2, pp. 227–244, 2000.

[117] M. Sugiyama, T. Suzuki, S. Nakajima, H. Kashima, P. Von Bünau, and M. Kawanabe, “Direct importance estimation for covariate shift adaptation,” *Annals of the Institute of Statistical Mathematics*, vol. 60, pp. 699–746, 2008.

[118] G. Dawson, J. M. Sun, K. S. Davlantis, M. Murias, L. Franz, J. Troy, R. Simmons, M. Sabatos-DeVito, R. Durham, and J. Kurtzberg, “Autologous cord blood infusions are safe and feasible in young children with autism spectrum disorder: results of a single-center phase i open-label trial,” *Stem cells translational medicine*, vol. 6, no. 5, pp. 1332–1339, 2017.

[119] J. A. Sáez and J. L. Romero-Béjar, “Impact of regressand stratification in dataset shift caused by cross-validation,” *Mathematics*, vol. 10, no. 14, p. 2538, 2022.

[120] Y. Li, M. Murias, S. Major, G. Dawson, and D. Carlson, “On target shift in adversarial domain adaptation,” in *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 616–625, PMLR, 2019.

[121] I. Redko, N. Courty, R. Flamary, and D. Tuia, “Optimal transport for multi-source domain adaptation under target shift,” in *The 22nd International Conference on artificial intelligence and statistics*, pp. 849–858, PMLR, 2019.

[122] J. von Kügelgen, L. Gresele, and B. Schölkopf, “Simpson’s paradox in covid-19 case fatality rates: a mediation analysis of age-related causal effects,” *IEEE Transactions on Artificial Intelligence*, vol. 2, no. 1, pp. 18–27, 2021.

[123] W. N. Evans and J. S. Ringel, “Can higher cigarette taxes improve birth outcomes?,” *Journal of public Economics*, vol. 72, no. 1, pp. 135–154, 1999.

[124] J. Mitrovic, B. McWilliams, J. Walker, L. Buesing, and C. Blundell, “Representation learning via invariant causal mechanisms,” *arXiv preprint arXiv:2010.07922*, 2020.

[125] Y. He, Z. Shen, and P. Cui, “Towards non-iid image classification: A dataset and baselines,” *Pattern Recognition*, vol. 110, p. 107383, 2021.

[126] D. Krueger, E. Caballero, J.-H. Jacobsen, A. Zhang, J. Binas, D. Zhang, R. Le Priol, and A. Courville, “Out-of-distribution generalization via risk extrapolation (rex),” in *International Conference on Machine Learning*, pp. 5815–5826, PMLR, 2021.

[127] J. Peters and P. Bühlmann, “Identifiability of gaussian structural equation models with equal error variances,” *Biometrika*, vol. 101, no. 1, pp. 219–228, 2014.

[128] A. Zanga, E. Ozkirimli, and F. Stella, “A survey on causal discovery: theory and practice,” *International Journal of Approximate Reasoning*, vol. 151, pp. 101–129, 2022.

[129] C. Glymour, K. Zhang, and P. Spirtes, “Review of causal discovery methods based on graphical models,” *Frontiers in genetics*, vol. 10, p. 524, 2019.

[130] S. Shimizu, P. O. Hoyer, A. Hyvärinen, A. Kerminen, and M. Jordan, “A linear non-gaussian acyclic model for causal discovery.,” *Journal of Machine Learning Research*, vol. 7, no. 10, 2006.

[131] C. MEEK, “Causal inference and causal explanation with background knowledge,” in *Proc. Conf. on Uncertainty in Artificial Intelligence (UAI-95)*, pp. 403–410, 1995.

[132] M. Learning, “Tom mitchell,” *Publisher: McGraw Hill*, 1997.

[133] C. Yeh, A. Perez, A. Driscoll, G. Azzari, Z. Tang, D. Lobell, S. Ermon, and M. Burke, “Using publicly available satellite imagery and deep learning to understand economic well-being in africa,” *Nature communications*, vol. 11, no. 1, p. 2583, 2020.

[134] A. Kendall and Y. Gal, “What uncertainties do we need in bayesian deep learning for computer vision?,” *Advances in neural information processing systems*, vol. 30, 2017.

[135] Y. Sun, A. K. Wong, and M. S. Kamel, “Classification of imbalanced data: A review,” *International journal of pattern recognition and artificial intelligence*, vol. 23, no. 04, pp. 687–719, 2009.

[136] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[137] Amazon.com, “Amazon sagemaker data labeling pricing.” `https://aws.amazon.com/sagemaker/data-labeling/pricing/`, visited 2023-03-31.

[138] I. Projec, “About imagenet.” `https://image-net.org/about.php`, visited 2023-03-31.

[139] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[140] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[141] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, “Mastering the game of go with deep neural networks and tree search,” *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[142] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, *et al.*, “A general reinforcement learning algorithm that masters chess, shogi, and go through self-play,” *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[143] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, and D. Amodei, “Deep reinforcement learning from human preferences,” *Advances in neural information processing systems*, vol. 30, 2017.

[144] OpenAI, “Gpt-4.” `https://openai.com/research/gpt-4`, visited 2023-03-31.

[145] L. Wasserman, *All of statistics: a concise course in statistical inference*, vol. 26. Springer, 2004.

[146] J. A. Thomas, “Elements of information theory,” 1991.

[147] A. Graves, G. Wayne, and I. Danihelka, “Neural turing machines,” *arXiv preprint arXiv:1410.5401*, 2014.

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

[149] K. Hornik, M. Stinchcombe, and H. White, “Multilayer feedforward networks are universal approximators,” *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[150] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function," *Neural networks*, vol. 6, no. 6, pp. 861–867, 1993.

[151] A. P. Dawid and M. Musio, "Theory and applications of proper scoring rules," *Metron*, vol. 72, no. 2, pp. 169–183, 2014.

[152] C. Lemaréchal, "Cauchy and the gradient method," *Doc Math Extra*, vol. 251, no. 254, p. 10, 2012.

[153] R. Grosse, "Csc321 lecture 6: Backpropagation." `https://www.cs.toronto.edu/~rgrosse/courses/csc321_2018/slides/lec06.pdf`, visited 2023-10-31.

[154] V. Vapnik, "Principles of risk minimization for learning theory," *Advances in neural information processing systems*, vol. 4, 1991.

[155] S. Richter, "Statistisches und maschinelles lernen," *Berlin, Heidelberg: Springer Berlin Heidelberg. DOI*, vol. 10, pp. 978–3, 2019.

[156] U. Von Luxburg and B. Schölkopf, "Statistical learning theory: Models, concepts, and results," in *Handbook of the History of Logic*, vol. 10, pp. 651–706, Elsevier, 2011.

[157] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, vol. 1, no. 1, pp. 67–82, 1997.

[158] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2. Springer, 2009.

[159] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning," *Advances in neural information processing systems*, vol. 20, 2007.

[160] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural computation*, vol. 4, no. 1, pp. 1–58, 1992.

[161] L. M. Bregman, "The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming," *USSR computational mathematics and mathematical physics*, vol. 7, no. 3, pp. 200–217, 1967.

[162] D. Pfau, "A generalized bias-variance decomposition for bregman divergences," *Unpublished Manuscript*, 2013.

[163] F. Nielsen and R. Nock, "Sided and symmetrized bregman centroids," *IEEE transactions on Information Theory*, vol. 55, no. 6, pp. 2882–2904, 2009.

[164] Anonymous, "Bias/variance is not the same as approximation/estimation," *Submitted to Transactions on Machine Learning Research*, 2023. Under review.

[165] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, "Deep double descent: Where bigger models and more data hurt," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2021, no. 12, p. 124003, 2021.

[166] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15849–15854, 2019.

[167] Curse of Dimensionality, "Curse of dimensionality — Wikipedia, the free encyclopedia." `https://en.wikipedia.org/wiki/Curse_of_dimensionality`, visited 2023-11-23.

[168] P. C. Kainen, "Utilizing geometric anomalies of high dimension: When complexity makes computation easier," 1997.

[169] 68–95–99.7 rule, "68–95–99.7 rule — Wikipedia, the free encyclopedia." `https://en.wikipedia.org/wiki/68%E2%80%9395%E2%80%9399.7_rule`, visited 2023-12-18.

[170] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[171] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[172] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*, pp. 2256–2265, PMLR, 2015.

[173] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in neural information processing systems*, vol. 32, 2019.

[174] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[175] C.-W. Huang, D. Krueger, A. Lacoste, and A. Courville, "Neural autoregressive flows," in *International Conference on Machine Learning*, pp. 2078–2087, PMLR, 2018.

[176] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," *arXiv preprint arXiv:1912.02762*, 2019.

[177] Y. Marzouk, T. Moselhy, M. Parno, and A. Spantini, "Sampling via measure transport: An introduction," in *Handbook of Uncertainty Quantification* (R. Ghanem, D. Higdon, and H. Owhadi, eds.), pp. 1–41, Springer, 2016.

[178] G. Papamakarios, E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan, "Normalizing flows for probabilistic modeling and inference," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 2617–2680, 2021.

[179] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, "Building machines that learn and think like people," *Behavioral and brain sciences*, vol. 40, 2017.

[180] E. Bareinboim and J. Pearl, "Causal inference and the data-fusion problem," *Proceedings of the National Academy of Sciences*, vol. 113, no. 27, pp. 7345–7352, 2016.

[181] A. Ghassami, S. Salehkaleybar, N. Kiyavash, and K. Zhang, "Learning causal structures using regression invariance," in *Advances in Neural Information Processing Systems*, pp. 3011–3021, 2017.

[182] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, "Analysis of representations for domain adaptation," in *Advances in neural information processing systems*, pp. 137–144, 2007.

[183] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, no. 2, pp. 199–210, 2010.

[184] D. Greenfeld and U. Shalit, "Robust learning with the hilbert-schmidt independence criterion," *arXiv preprint arXiv:1910.00270*, 2019.

[185] C. Xie, F. Chen, Y. Liu, and Z. Li, "Risk variance penalization: From distributional robustness to causality," *arXiv preprint arXiv:2006.07544*, 2020.

[186] K. Zhang, M. Gong, and B. Schölkopf, "Multi-source domain adaptation: A causal view," in *Twenty-ninth AAAI conference on artificial intelligence*, 2015.

[187] P. Spirtes and C. Glymour, "An algorithm for fast recovery of sparse causal graphs," *Social science computer review*, vol. 9, no. 1, pp. 62–72, 1991.

[188] D. M. Chickering, "Optimal structure identification with greedy search," *Journal of machine learning research*, vol. 3, no. Nov, pp. 507–554, 2002.

[189] K. D. Hoover, "The logic of causal inference: Econometrics and the conditional analysis of causation," *Economics & Philosophy*, vol. 6, no. 2, pp. 207–234, 1990.

[190] J. Tian and J. Pearl, "Causal discovery from changes," *Uncertainty in Artificial Intelligence (UAI)*, pp. 512–521, 2001.

[191] A. Ghassami, N. Kiyavash, B. Huang, and K. Zhang, "Multi-domain causal structure learning in linear systems," in *Advances in neural information processing systems*, pp. 6266–6276, 2018.

[192] B. Huang, K. Zhang, J. Zhang, J. Ramsey, R. Sanchez-Romero, C. Glymour, and B. Schölkopf, "Causal discovery from heterogeneous/nonstationary data," *Journal of Machine Learning Research*, vol. 21, no. 89, pp. 1–53, 2020.

[193] R. Frisch, "Statistical versus theoretical relations in economic macrodynamics.paper given at league of nations. reprinted in d.f. hendry and m.s. morgan (1995)," *The Foundations of Econometric Analysis*, 1938.

[194] J. J. Heckman and R. Pinto, "Causal analysis after haavelmo," tech. rep., National Bureau of Economic Research, 2013.

[195] C. Heinze-Deml, J. Peters, and N. Meinshausen, "Invariant causal prediction for nonlinear models," *Journal of Causal Inference*, vol. 6, no. 2, 2018.

[196] L. Ardizzone, C. Lüth, J. Kruse, C. Rother, and U. Köthe, "Guided image generation with conditional invertible neural networks," *arXiv preprint arXiv:1907.02392*, 2019.

[197] J. Peters, J. M. Mooij, D. Janzing, and B. Schölkopf, "Causal discovery with continuous additive noise models," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 2009–2053, 2014.

[198] A. Gretton, O. Bousquet, A. Smola, and B. Schölkopf, "Measuring statistical dependence with hilbert-schmidt norms," in *International conference on algorithmic learning theory*, pp. 63–77, Springer, 2005.

[199] Z. Qin and D. Kim, "Rethinking softmax with cross-entropy: Neural network classifier as mutual information estimator," *arXiv preprint arXiv:1911.10688*, 2019.

[200] D. Barber and F. V. Agakov, "The im algorithm: a variational approach to information maximization," in *Advances in neural information processing systems*, p. None, 2003.

[201] D. Kalainathan, O. Goudet, I. Guyon, D. Lopez-Paz, and M. Sebag, “Sam: Structural agnostic model, causal discovery and penalized adversarial learning,” *arXiv preprint arXiv:1803.04929*, 2018.

[202] L. Matthey, I. Higgins, D. Hassabis, and A. Lerchner, “dsprites: Disentanglement testing sprites dataset.” https://github.com/deepmind/dsprites-dataset/, 2017.

[203] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, “Virtual worlds as proxy for multi-object tracking analysis,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4340–4349, 2016.

[204] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “Carla: An open urban driving simulator,” in *Conference on robot learning*, pp. 1–16, PMLR, 2017.

[205] H. Edwards and A. Storkey, “Towards a neural statistician,” 2017.

[206] B. Bloem-Reddy and Y. W. Teh, “Probabilistic symmetries and invariant neural networks,” *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 3535–3595, 2020.

[207] J. Yang, P. Wang, D. Zou, Z. Zhou, K. Ding, W. Peng, H. Wang, G. Chen, B. Li, Y. Sun, *et al.*, “Openood: Benchmarking generalized out-of-distribution detection,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 32598–32611, 2022.

[208] P. Orbanz and D. M. Roy, “Bayesian models of graphs, arrays and other exchangeable random structures,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 2, pp. 437–461, 2014.

[209] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, “Deep sets,” *Advances in neural information processing systems*, vol. 30, 2017.

[210] E. Wagstaff, F. B. Fuchs, M. Engelcke, M. A. Osborne, and I. Posner, “Universal approximation of functions on sets,” *Journal of Machine Learning Research*, vol. 23, no. 151, pp. 1–56, 2022.

[211] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh, “Set transformer: A framework for attention-based permutation-invariant neural networks,” 2019.

[212] J. Liang, R. He, and T. Tan, “A comprehensive survey on test-time adaptation under distribution shifts,” *arXiv preprint arXiv:2303.15361*, 2023.

[213] Y. Sun, X. Wang, Z. Liu, J. Miller, A. Efros, and M. Hardt, “Test-time training with self-supervision for generalization under distribution shifts,” in *International conference on machine learning*, pp. 9229–9248, PMLR, 2020.

[214] M. Zhang, S. Levine, and C. Finn, “Memo: Test time robustness via adaptation and augmentation,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 38629–38642, 2022.

[215] S. Schneider, E. Rusak, L. Eck, O. Bringmann, W. Brendel, and M. Bethge, “Improving robustness against common corruptions by covariate shift adaptation,” *Advances in neural information processing systems*, vol. 33, pp. 11539–11551, 2020.

[216] M. Schmitt, P.-C. Bürkner, U. Köthe, and S. T. Radev, “Detecting model misspecification in amortized bayesian inference with neural networks,” *arXiv preprint arXiv:2112.08866*, 2021.

[217] E. Creager, J.-H. Jacobsen, and R. Zemel, “Environment inference for invariant learning,” in *International Conference on Machine Learning*, pp. 2189–2200, PMLR, 2021.

[218] R. L. Murphy, B. Srinivasan, V. Rao, and B. Ribeiro, “Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs,” *arXiv preprint arXiv:1811.01900*, 2018.

[219] S. Zare and H. Van Nguyen, “Picaso: Permutation-invariant cascaded attentional set operator,” *arXiv preprint arXiv:2107.08305*, 2021.

[220] D. dan Guo, L. Tian, M. Zhang, M. Zhou, and H. Zha, “Learning prototype-oriented set representations for meta-learning,” in *International Conference on Learning Representations*, 2021.

[221] S. Bartunov, F. B. Fuchs, and T. P. Lillicrap, “Equilibrium aggregation: encoding sets via optimization,” in *Uncertainty in Artificial Intelligence*, pp. 139–149, PMLR, 2022.

[222] C. K. Wikle, “Hierarchical bayesian models for predicting the spread of ecological processes,” *Ecology*, vol. 84, no. 6, pp. 1382–1394, 2003.

[223] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin, *Bayesian Data Analysis (3rd edition)*. Chapman and Hall/CRC, 2013.

[224] H. Edwards and A. Storkey, “Towards a neural statistician,” *arXiv preprint arXiv:1606.02185*, 2016.

[225] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. Eslami, and Y. W. Teh, “Neural processes,” *arXiv preprint arXiv:1807.01622*, 2018.

[226] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh, “Attentive neural processes,” *arXiv preprint arXiv:1901.05761*, 2019.

[227] C. E. Rasmussen, “Gaussian processes in machine learning,” in *Summer school on machine learning*, pp. 63–71, Springer, 2003.

[228] J. Kim, J. Yoo, J. Lee, and S. Hong, “Setvae: Learning hierarchical composition for generative modeling of set-structured data,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15059–15068, 2021.

[229] X. Zeng, A. Vahdat, F. Williams, Z. Gojcic, O. Litany, S. Fidler, and K. Kreis, “Lion: Latent point diffusion models for 3d shape generation,” *arXiv preprint arXiv:2210.06978*, 2022.

[230] C. C. Aggarwal and P. S. Yu, “Outlier detection for high dimensional data,” in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pp. 37–46, 2001.

[231] S. Han, X. Hu, H. Huang, M. Jiang, and Y. Zhao, “Adbench: Anomaly detection benchmark,” *arXiv:2206.09426*, 2022.

[232] J. Yang, P. Wang, D. Zou, Z. Zhou, K. Ding, W. Peng, H. Wang, G. Chen, B. Li, Y. Sun, *et al.*, “Openood: Benchmarking generalized out-of-distribution detection,” *arXiv:2210.07242*, 2022.

[233] K. Hendrickx, L. Perini, D. Van der Plas, W. Meert, and J. Davis, “Machine learning with a reject option: A survey,” *arXiv preprint arXiv:2107.11277*, 2021.

[234] M. E. Hellman, “The nearest neighbor classification rule with a reject option,” *IEEE Transactions on Systems Science and Cybernetics*, vol. 6, no. 3, pp. 179–185, 1970.

[235] G. Fumera and F. Roli, "Support vector machines with embedded reject option," in *Pattern Recognition with Support Vector Machines: First International Workshop, SVM 2002 Niagara Falls, Canada, August 10, 2002 Proceedings*, pp. 68–82, Springer, 2002.

[236] Y. Grandvalet, A. Rakotomamonjy, J. Keshet, and S. Canu, "Support vector machines with a reject option," *Advances in neural information processing systems*, vol. 21, 2008.

[237] M. Wegkamp and M. Yuan, "Support vector machines with a reject option," *arXiv preprint arXiv:1201.1140*, 2012.

[238] C.-K. Chow, "An optimum character recognition system using decision functions," *IRE Transactions on Electronic Computers*, pp. 247–254, 1957.

[239] C. Chow, "On optimum recognition error and reject tradeoff," *IEEE Transactions on information theory*, vol. 16, no. 1, pp. 41–46, 1970.

[240] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *arXiv:1610.02136*, 2016.

[241] D. Hendrycks, S. Basart, M. Mazeika, M. Mostajabi, J. Steinhardt, and D. Song, "Scaling out-of-distribution detection for real-world settings," *arXiv:1911.11132*, 2019.

[242] W. Liu, X. Wang, J. Owens, and Y. Li, "Energy-based out-of-distribution detection," *Advances in neural information processing systems*, vol. 33, pp. 21464–21475, 2020.

[243] C. C. Aggarwal and C. C. Aggarwal, *An introduction to outlier analysis*. Springer, 2017.

[244] Y. Sun, Y. Ming, X. Zhu, and Y. Li, "Out-of-distribution detection with deep nearest neighbors," in *International Conference on Machine Learning*, pp. 20827–20840, PMLR, 2022.

[245] H. Wang, Z. Li, L. Feng, and W. Zhang, "Vim: Out-of-distribution with virtual-logit matching," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4921–4930, 2022.

[246] Wikipedia, The Free Encyclopedia, "Simpson's paradox." `https://en.wikipedia.org/wiki/Simpson%27s_paradox#/media/File:Simpsons_paradox_-_animation.gif`, visited 2023-12-12.

[247] H. Fanaee-T, "Bike Sharing Dataset." UCI Machine Learning Repository, 2013. DOI: https://doi.org/10.24432/C5W894.

[248] A. Sanner, C. Gonzalez, and A. Mukhopadhyay, "How reliable are out-of-distribution generalization methods for medical image segmentation?," in *DAGM German Conference on Pattern Recognition*, pp. 604–617, Springer, 2021.

[249] I. Flores, "An optimum character recognition system using decision functions," *IRE Transactions on Electronic Computers*, pp. 180–180, 1958.

[250] X.-Y. Zhang, G.-S. Xie, X. Li, T. Mei, and C.-L. Liu, "A survey on learning to reject," *Proceedings of the IEEE*, 2023.

[251] J. Xie, Z. Qiu, and J. Wu, "Bootstrap methods for reject rules of fisher lda," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 3, pp. 425–428, IEEE, 2006.

[252] P. L. Bartlett and M. H. Wegkamp, "Classification with a reject option using a hinge loss.," *Journal of Machine Learning Research*, vol. 9, no. 8, 2008.

[253] E. Techapanurak and T. Okatani, “Practical evaluation of out-of-distribution detection methods for image classification,” *arXiv:2101.02447*, 2021.

[254] G. Xia and C.-S. Bouganis, “Augmenting softmax information for selective classification with out-of-distribution data,” in *Proceedings of the Asian Conference on Computer Vision*, pp. 1995–2012, 2022.

[255] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Müller, “A unifying review of deep and shallow anomaly detection,” *Proceedings of the IEEE*, vol. 109, no. 5, pp. 756–795, 2021.

[256] L. A. Gatys, A. S. Ecker, and M. Bethge, “Image style transfer using convolutional neural networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2414–2423, 2016.

[257] Z. Xiaojin, “Semi-supervised learning literature survey,” *Computer Sciences TR*, vol. 1530, 2008.

[258] L. Chen, Y. Zhang, Y. Song, Y. Shan, and L. Liu, “Improved test-time adaptation for domain generalization,” *arXiv preprint arXiv:2304.04494*, 2023.

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

[260] E. Nalisnick, A. Matsukawa, Y. W. Teh, and B. Lakshminarayanan, “Detecting out-of-distribution inputs to deep generative models using typicality,” *arXiv:1906.02994*, 2019.

[261] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira, “Analysis of representations for domain adaptation,” *Advances in neural information processing systems*, vol. 19, 2006.

[262] Y. Shi, J. Seely, P. H. Torr, N. Siddharth, A. Hannun, N. Usunier, and G. Synnaeve, “Gradient matching for domain generalization,” *arXiv:2104.09937*, 2021.

[263] S. Korevaar, R. Tennakoon, and A. Bab-Hadiashar, “Failure to achieve domain invariance with domain generalization algorithms: An analysis in medical imaging,” *IEEE Access*, 2023.

[264] A. Kamath, R. Jia, and P. Liang, “Selective question answering under domain shift,” *arXiv:2006.09462*, 2020.

[265] N. Varshney, S. Mishra, and C. Baral, “Investigating selective prediction approaches across several tasks in iid, ood, and adversarial settings,” *arXiv:2203.00211*, 2022.

[266] J. Ren, J. Luo, Y. Zhao, K. Krishna, M. Saleh, B. Lakshminarayanan, and P. J. Liu, “Out-of-distribution detection and selective generation for conditional language models,” *arXiv:2209.15558*, 2022.

[267] D. P. Mesquita, L. S. Rocha, J. P. P. Gomes, and A. R. R. Neto, “Classification with reject option for software defect prediction,” *Applied Soft Computing*, vol. 49, pp. 1085–1093, 2016.

[268] F. Condessa, J. Bioucas-Dias, and J. Kovačević, “Performance measures for classification systems with rejection,” *Pattern Recognition*, vol. 63, pp. 437–450, 2017.

[269] M. S. A. Nadeem, J.-D. Zucker, and B. Hanczar, “Accuracy-rejection curves (arcs) for comparing classification methods with a reject option,” in *Machine Learning in Systems Biology*, pp. 65–81, PMLR, 2009.

[270] Y. Wiener and R. El-Yaniv, “Agnostic selective classification,” *Advances in neural information processing systems*, vol. 24, 2011.

[271] K. Lee, K. Lee, H. Lee, and J. Shin, “A simple unified framework for detecting out-of-distribution samples and adversarial attacks,” *Advances in neural information processing systems*, vol. 31, 2018.

[272] Y. Sun, C. Guo, and Y. Li, “React: Out-of-distribution detection with rectified activations,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 144–157, 2021.

[273] S. Sagawa, P. W. Koh, T. B. Hashimoto, and P. Liang, “Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization,” *arXiv:1911.08731*, 2019.

[274] M. Pezeshki, O. Kaba, Y. Bengio, A. C. Courville, D. Precup, and G. Lajoie, “Gradient starvation: A learning proclivity in neural networks,” *Advances in Neural Information Processing Systems*, vol. 34, pp. 1256–1272, 2021.

[275] S. Yan, H. Song, N. Li, L. Zou, and L. Ren, “Improve unsupervised domain adaptation with mixup training,” *arXiv:2001.00677*, 2020.

[276] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[277] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *arXiv:2010.11929*, 2020.

[278] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, “Swin transformer: Hierarchical vision transformer using shifted windows,” in *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 10012–10022, 2021.

[279] S. D. D. Cruz, O. Wasenmuller, H.-P. Beise, T. Stifter, and D. Stricker, “Sviro: Synthetic vehicle interior rear seat occupancy dataset and benchmark,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 973–982, 2020.

[280] R. Stephen and B. Anthony, *Hierarchical linear models.* Sage Publications, Thousand Oaks, CA, 2002.

[281] C. Louizos, M. Welling, and D. P. Kingma, “Learning sparse neural networks through l_0 regularization,” *arXiv preprint arXiv:1712.01312*, 2017.

[282] S. Kolouri, P. E. Pope, C. E. Martin, and G. K. Rohde, “Sliced-wasserstein autoencoder: An embarrassingly simple generative model,” *arXiv preprint arXiv:1804.01947*, 2018.

[283] P. J. Bickel, E. A. Hammel, and J. W. O’Connell, “Sex bias in graduate admissions: Data from berkeley: Measuring bias is harder than is usually assumed, and the evidence is sometimes contrary to expectation.,” *Science*, vol. 187, p. 398–404, Feb. 1975.

[284] C. R. Charig, D. R. Webb, S. R. Payne, and J. E. Wickham, “Comparison of treatment of renal calculi by open surgery, percutaneous nephrolithotomy, and extracorporeal shockwave lithotripsy.,” *BMJ*, vol. 292, p. 879–882, Mar. 1986.

[285] A. F. Agarap, "Deep learning using rectified linear units (relu)," 2019.

[286] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," 2018.

[287] D. Rothenhäusler, N. Meinshausen, P. Bühlmann, and J. Peters, "Anchor regression: Heterogeneous data meet causality," *Journal of the Royal Statistical Society Series B: Statistical Methodology*, vol. 83, no. 2, pp. 215–246, 2021.

[288] F. Wenzel, A. Dittadi, P. V. Gehler, C.-J. Simon-Gabriel, M. Horn, D. Zietlow, D. Kernert, C. Russell, T. Brox, B. Schiele, *et al.*, "Assaying out-of-distribution generalization in transfer learning," *arXiv preprint arXiv:2207.09239*, 2022.

Appendix: Causality and Learning with the Principle of ICM

A

A.1. Instrumental Variables and Hidden Confounders

Recall that the SCM in the example in Section 3.8 is described as

$$X := bE + cH + N_X \tag{A.1}$$
$$Y := aX + dH + N_Y \tag{A.2}$$

where N_X, N_Y and $H = N_H$ is jointly independent and unobserved noise. It is assumed that $b, c, d \neq 0$ and that $\mathrm{Var}(H) \neq 0$.

If we just regress on X to predict Y, we obtain an estimate for a that converges in the infinite data regime to

$$\widetilde{a} = \frac{\mathrm{Cov}(X,Y)}{\mathrm{Var}(X)} = \frac{a \cdot \mathrm{Var}(X) + d \cdot \mathrm{Cov}(X,H)}{\mathrm{Var}(X)} \tag{A.3}$$
$$= \frac{a\,\mathrm{Var}(X) + d \cdot c \cdot \mathrm{Var}(H)}{\mathrm{Var}(X)} = a + \frac{d \cdot c \cdot \mathrm{Var}(H)}{\mathrm{Var}(X)} \tag{A.4}$$

This is obviously a biased estimate. For simplicity, we assume in the following that all variables have zero expectancy. If we use this biased estimate for prediction, we obtain

$$\mathbb{E}_{X,Y}[(\widetilde{a}X - Y)^2] = \mathbb{E}\left((\widetilde{a} - a)X - dH - N_Y\right)^2 \tag{A.5}$$
$$= \mathbb{E}[(\widetilde{a} - a^2)X^2] - 2\mathbb{E}[(\widetilde{a} - a)X \cdot (dH + N_Y)] - \mathbb{E}(dH + N_Y)^2 \tag{A.6}$$
$$= \mathbb{E}[(\widetilde{a} - a)^2 X^2] - 2\mathbb{E}[(\widetilde{a} - a) \cdot d \cdot XH] - \mathbb{E}(dH + N_Y)^2 \tag{A.7}$$
$$= \frac{d^2c^2\,\mathrm{Var}(H)^2}{\mathrm{Var}(X)^2}\mathrm{Var}(X) - 2\frac{d^2c^2\,\mathrm{Var}(H)}{\mathrm{Var}\,X}\mathrm{Var}(H) + \mathbb{E}(dH + N_Y)^2 \tag{A.8}$$
$$= -\frac{d^2c^2\,\mathrm{Var}(H)^2}{\mathrm{Var}(X)} + \mathbb{E}(dH + N_Y)^2 < \mathbb{E}(dH + N_Y)^2 \tag{A.9}$$

Here we primarily utilize that $N_Y \perp X$, $E \perp H$, $N_X \perp H$ and the fact that $\mathbb{E}[A \cdot B] = 0$ holds for independent RVs A and B. A model based on $\widetilde{a}$ achieves a smaller predictive loss in Equation A.9 compared to the one attained by the causal model:

$$\mathbb{E}(aX - Y)^2 = \mathbb{E}(dH + N_Y)^2 = \mathbb{E}[d^2H^2] + \mathbb{E}[N_Y^2] \tag{A.10}$$

Conclusively, we would prefer model $\widetilde{f}(X) = \widetilde{a}X$ over $f(X) = aX$ in terms of predictive loss. However, $\widetilde{f}$ is not as robust to environment changes. Consider the case where the environment $E = 0$ eradicates the effect of H on X, i.e. $X := N_X$. In this case, the biased predictor $\widetilde{f}$ introduces an unnecessary bias:

$$\begin{aligned}
\mathbb{E}(\widetilde{a}X - Y)^2 &= \mathbb{E}\left((\widetilde{a} - a)X - dH - N_Y\right)^2 && \text{(A.11)}\\
&= \mathbb{E}[(\widetilde{a} - a)^2X^2] - 2\mathbb{E}[(\widetilde{a} - a)X \cdot (dH + N_Y)] + \mathbb{E}(dH + N_Y)^2 && \text{(A.12)}\\
&= \mathbb{E}[(\widetilde{a} - a)^2X^2] + \mathbb{E}(dH + N_Y)^2 && \text{(A.13)}\\
& && \text{(A.14)}
\end{aligned}$$

The causal model that only uses the "pure" causal effect from X to predict Y is robust with respect to environment changes and attains a smaller predictive loss:

$$\mathbb{E}(aX - Y)^2 = \mathbb{E}(dH + N_Y)^2 \tag{A.15}$$

So how could we estimate the "pure" effect of X on Y? One way to do this is to use *instrumental variables* (IVs). An instrumental variable (IV) is a variable that is (i) independent of the hidden confounder H, (ii) dependent on X and (iii) affects Y only through X [32, Chapter 9.3]. An instrumental variable can also be interpreted as an environment variable. We show a graph that corresponds to the requirements of an IV in Figure 3.11. Since E is independent of H and N_X, we can consider $cH + N_X$ as noise

$$X := bE + (cH + N_X) \tag{A.16}$$

Hence, we can consistently estimate b without introducing any biases. The target variable then becomes

$$Y := aX + dH + N_Y = a(bE) + [a(cH + N_X) + dH + N_Y] \tag{A.17}$$

since bE is independent of the noise $a(cH + N_X) + dH + N_Y$, we can consistently estimate a by regressing on bE. This two-stage procedure is also called *two-stage least squares*.

A.2. Learning with the Principle of ICM

A.2.1. Gating Architecture

We employ the same gating architecture as in [201] which was first proposed in [281] as a Bernoulli reparameterization trick. They use this reparameterization trick in their original work in order to train neural networks with L0-Regularization in a gradient based manner. [201] apply the L0-Regularization on the input to learn a gating mechanism. Similarly we use the L0-Regularization to learn a gating mechanism.

The gating architecture h_ϕ is parameterized via $\phi = (\boldsymbol{\alpha}, \boldsymbol{\beta})$ where $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_D)$ and $\boldsymbol{\beta} = (\beta_1, \dots, \beta_D)$. Let $\gamma < 0$ and $\zeta > 0$ be fixed. Then we map $\boldsymbol{u} \sim \mathcal{U}[0,1]^D$ via $\boldsymbol{s}(\boldsymbol{u}) = \text{Sigmoid}((\log \boldsymbol{u} - \log(1-\boldsymbol{u}) + \boldsymbol{\alpha})/\boldsymbol{\beta})$, to $\boldsymbol{z} = \min(1, \max(0, \boldsymbol{s}(\boldsymbol{u})(\zeta - \gamma) + \gamma))$. This is how we sample the gates for each batch during training. The gates are then multiplied element-wise with the input $\boldsymbol{z} \odot \mathbf{X}$. In principle we could sample many $u \sim \mathcal{U}[0,1]$, but we observe that one sample of $u \sim \mathcal{U}[0,1]$ per batch suffices for our examples. At test time we use the following estimator for the gates:

$$\hat{\boldsymbol{z}} = \min(1, \max(0, \text{Sigmoid}(\boldsymbol{\alpha})(\zeta - \gamma) + \gamma))$$

Similarly as during training time, we multiply $\hat{\boldsymbol{z}}$ with the input. After sufficient training $\hat{\boldsymbol{z}}$ is a hard 0-1 mask. The complexity loss is defined via

$$\mathcal{L}(h_{\boldsymbol{\theta}}) = \sum_{j=1}^{D} \text{Sigmoid}\left(\alpha_j - \beta_j \log \frac{-\gamma}{\zeta}\right). \tag{A.18}$$

For a detailed derivation of the reparameterization and complexity loss, see [281].

A.2.2. Wasserstein Loss

The one dimensional Wasserstein loss compares the similarity of two distributions [282]. This loss has expectation 0 if both distributions are equal. An empirical estimate of the one dimensional Wasserstein loss for two random variables A, B is given by

$$\mathcal{L}_W = \|\text{sort}(\{a_j\}_{j=1}^n) - \text{sort}(\{b_j\}_{j=1}^n)\|_2$$

Here, the two batches are sorted in ascending order and then compared in the L2-Norm. We assume that both batches have the same size.

A.3. Experimental Setting for Synthetic Dataset

A.3.1. Data Generation

In Section 5.5 we described how we choose different Structural Causal Models (SCM). In the following we describe details of this process.

We simulate the datasets in a way that the conditions in Proposition 7 are met. We choose different variables in the graph shown in Figure 5.5 as target variable. Hence, we consider different “topological” scenarios. We assume the data is generated by some underlying SCM. We define the structural assignments in the SCM as follows

$$\begin{aligned}
&\text{(a)}\quad f_i^{(1)}(\mathbf{X}_{pa(i)}, N_i) = \sum_{j \in pa(i)} a_j X_j + N_i \quad \text{[Linear]}\\
&\text{(b)}\quad f_i^{(2)}(\mathbf{X}_{pa(i)}, N_i) = \sum_{j \in pa(i)} a_j X_j - \tanh(a_j X_j) + N_i\\
&\qquad \text{[Tanhshrink]}\\
&\text{(c)}\quad f_i^{(3)}(\mathbf{X}_{pa(i)}, N_i) = \sum_{j \in pa(i)} \log(1 + \exp(a_j X_j)) + N_i\\
&\qquad \text{[Softplus]}\\
&\text{(d)}\quad f_i^{(4)}(\mathbf{X}_{pa(i)}, N_i) = \sum_{j \in pa(i)} \max\{0, a_j X_j)\} + N_i\\
&\qquad \text{[ReLU]}\\
&\text{(e)}\quad f_i^{(5)}(\mathbf{X}_{pa(i)}, N_i) = \Big(\sum_{j \in pa(i)} a_j X_j \Big) \cdot (1 + \frac{1}{4} N_i) + N_i\\
&\qquad \text{[Mult. Noise]}
\end{aligned}$$

with $N_i \sim \mathcal{N}(0, c_i^2)$ where $c_i \sim \mathcal{U}[0.8, 1.2], i \in \{0, \dots, 5\}$ and $a_i \in \{-1, 1\}$ according to Figure A.1. Note that the mechanisms in (b), (c) and (d) are non-linear with additive noise and (e) elaborates the noise in a non-linear manner.

We consider hard- and soft-interventions on the assignments f_i. We either intervene on all variables except the target variable at once *or* on all parents and children of the target variable (Intervention Location). We consider three types of interventions:

- *Hard-Intervention* on X_i: Force $X_i \sim e_1 + e_2 \mathcal{N}(0, 1)$ where we sample for each environment $e_2 \sim \mathcal{U}([1.5, 2.5])$ and $e_1 \sim \mathcal{U}([0.5, 1.5] \cup [-1.5, -0.5])$
- *Soft-Intervention* I on X_i: Add $e_1 + e_2 \mathcal{N}(0, 1)$ to X_i where we sample for each environment $e_2 \sim \mathcal{U}([1.5, 2.5])$ and $e_1 \sim \mathcal{U}([0.5, 1.5] \cup [-1.5, -0.5])$
- *Soft-Intervention* II on X_i: Set the noise distribution N_i to $\mathcal{N}(0, 2^2)$ for $E = 2$ and to $\mathcal{N}(0, 0.2^2)$ for $E = 3$

Per run, we consider one environment without intervention ($E = 1$) and two environments with either both soft- or hard-interventions ($E = 2, 3$). We also create a fourth environment to measure a models' ability for out-of-distribution generalization:

- *Hard-Intervention*: Force $X_i \sim e + \mathcal{N}(0, 4^2)$ where $e = e_1 \pm 1$ with e_1 from environment $E = 1$. The sign $\{+, -\}$ is chosen once for each i with equal probability.
- *Soft-Intervention* I: Add $e + \mathcal{N}(0, 4^2)$ to X_i where $e = e_1 \pm 1$ with e_1 from environment $E = 1$. The sign $\{+, -\}$ is chosen once for each i with equal probability as for the *do-intervention* case.
- *Soft-Intervention* II: Half of the samples have noise N_i distributed due to $\mathcal{N}(0, 1.2^2)$ and the other half of the samples have noise distributed as $\mathcal{N}(0, 3^2)$

We randomly sample causal graphs as described above. Per environment, we consider 1024 samples.

A.3.2. Training Details

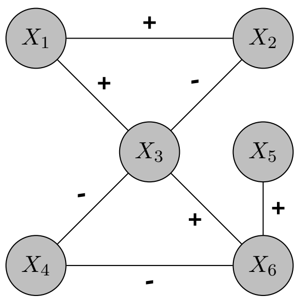


Figure A.1. The signs of the coefficients a_j for the mechanisms of the different SCMs

All used feed forward neural networks have two internal layers of size 256. For the normalizing flows we use a 2 layer *MTA-Flow* described in Appendix A.3.3 with K=32. As optimizer we use Adam with a learning rate of 10^{-3} and a L2-Regularizer weighted by 10^{-5} for all models. Each model is trained with a batch size of 256. We train each model for 1000 epochs and decay the learning rate every 400 epochs by 0.5. For each model we use $\lambda_I = 256$ and the HSIC $\mathcal{L}_I$ employs a Gaussian kernel with $\sigma = 1$. The gating architecture was trained without the complexity loss for 200 epochs and then with complexity loss weighted by 5. For the Flow model without gating architecture we use a feed forward neural network h_ϕ with two internal layers of size 256 mapping to an one dimensional vector. In total, we evaluated our models on 1365 created datasets as described in Appendix A.3.1.

Once the normalizing flow T is learned, we predict y given features $h(\mathbf{x})$ using 512 normally distributed samples u_i which are mapped to samples from $p(y|h(\mathbf{x}))$ by the trained normalizing flow $T(u_i; h(\mathbf{x}))$. As prediction we use the mean of these samples.

A.3.3. One-Dimensional Normalizing Flow

We use as one-dimension normalizing flow the *More-Than-Affine-Flow (MTA-Flow)*, which was developd by us. An overview of different architectures for one-dimensional normalizing flows can be found in [176]. For each layer of the flow, a conditioner network C maps the conditional data $h(\mathbf{X})$ to a set of parameters $a, b \in \mathbb{R}$ and $\mathbf{w}, \mathbf{v}, \mathbf{r} \in \mathbb{R}^K$ for a chosen $K \in \mathbb{N}$. It builds the transformer τ for each layer as

$$\begin{aligned} z &= \tau(y \mid h(\mathbf{X})) \\ &:= a\left(y + \frac{1}{N(\mathbf{w},\mathbf{v})}\sum_{i=1}^{K} w_i f(v_i y + r_i)\right) + b, \end{aligned} \tag{A.19}$$

where f is any almost everywhere smooth function with a derivative bounded by 1. In this work we used a gaussian function with normalized derivative for f. The division by

$$N(\mathbf{w},\mathbf{v}) := \varepsilon^{-1}\left(\sum_{i=1}^{K} |w_i v_i| + \delta\right), \tag{A.20}$$

with numeric stabilizers $\varepsilon < 1$ and $\delta > 0$, assures the strict monotonicity of τ and thus its invertibility $\forall x \in \mathbb{R}$. We also used a slightly different version of the *MTA-Flow* which uses the ELU activation function and – because of its monotonicity – can use a relaxed normalizing expression $N(\mathbf{w}, \mathbf{v})$.

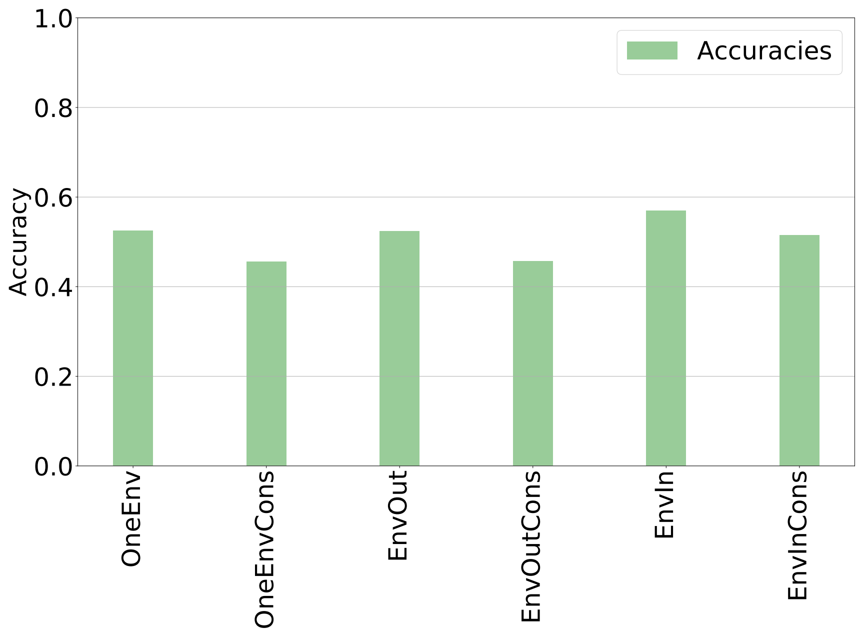


Figure A.2. Detection accuracies of direct causes for different variants of the PC-Algorithm. EnvOut means we pool over all environments and EnvIn means the environment is treated as system intern variable E. The suffix Cons means we us the conservative assignment scheme. OneEnv means we only consider the observational environment for inference.

A.3.4. PC-Variant

Since we are interested in the direct causes of Y, the widely applied PC-Algorithm gives not the complete answer to the query for the parents of Y. This is due to the fact that it is not able to orient all edges. To compare the PC-Algorithm we include the environment as system-intern variable and use a conservative assignment scheme where non-oriented edges are thrown away. This assignment scheme corresponds to the conservative nature of the ICP.

For further interest going beyond this work, we consider diverse variants of the PC-Algorithm. We consider two orientation schemes: A *conservative* one, where non-oriented edges are thrown away and a *non-conservative* one where non-oriented edges from a node X_i to Y are considered parents of Y.

We furthermore consider three scenarios: (1) the samples across all environments are pooled, (2) only the observational data (from the first environment) is given, and (3) the environment variable is considered as system-intern variable and is seen by the PC-Algorithm (similar as in [93]). Results are shown in Figure A.2. In order to obtain these results, we sampled 1500 graphs as described above and applied on each of these datasets a PC-Variant. Best accuracies are achieved if we consider the environment variable as system-intern variable and use the non-conservative orientation scheme (EnvIn).

A.3.5. Variable Selection

We consider the task of finding the direct causes of a target variable Y. Our models based on the gating mechanism perform a variable selection and are therefore compared to the

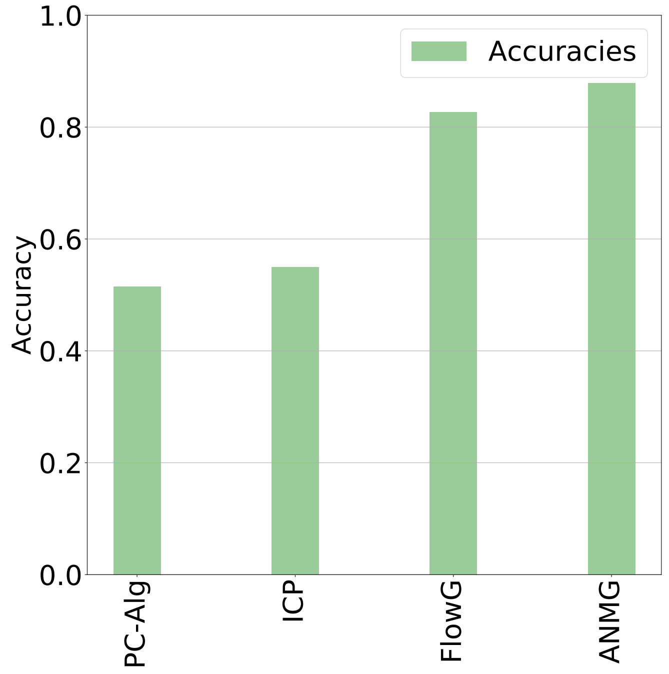


Figure A.3. Accuracies for different models across all scenarios. FlowG and ANMG are our models.

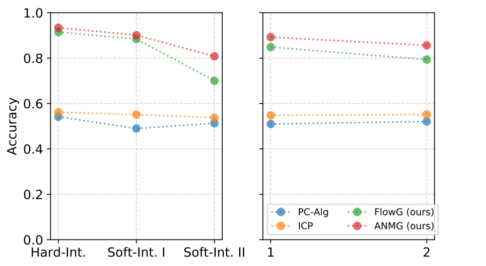
PC-Algorithm and ICP. In the following, we show the accuracies of this variable selection according to different scenarios.

Figure A.3 shows the accuracies of ICP, the PC-Algorithm and our models pooled over all scenarios. Our models perform comparably well and better than the baseline in the causal discovery task.

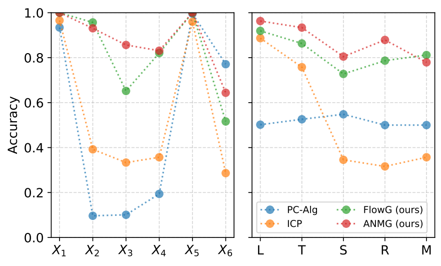
In the following, we show results due to different mechanisms, target variables, intervention types, and intervention locations. Figure A.4b shows the accuracies of all models across different target variables. Parentless target variables, i.e. $Y = X_4$ or $Y = X_0$ are easy to solve for ICP due to its conservative nature. All our models solve the parentless case quite well. The performance of the PC-variant depends strongly on the position of the target variable in the SCM indicating that its conservative assignment scheme has a strong influence on its performance. As expected, the PC-variant deals well with $Y = X_6$ which is a childless collider. The causal discovery task seems to be particularly hard for variable $Y = X_6$ for all other models. This is the variable which has the most parents.

The type of intervention and its location seem to play a minor role as shown in Figure A.4a and Figure A.4a.

Figure A.4b shows that ICP performs well if the underlying causal model is linear, but degrades if the mechanism become non-linear. The PC-Algorithm performs under all mechanisms comparably, but not well. ANMG performs quite well in all cases and even slightly better than FlowG in the cases of additive noise. However in the case of non-additive noise FlowG performs quite well whereas ANMG perform slightly worse – arguably because their requirements (additive noise) on the underlying mechanisms are not met.



(a) Accuracies of different models for different intervention types and locations. 1 stands for intervention on all variables except Y and 2 stands for interventions on parents and children only.



(b) Accuracies of different models according to target variables and mechanisms of the underlying SCM. Here we compare the PC-Alg, ICP with our models FlowG and ANMG that both employ a gating mechanism as feature.

Figure A.4. Comparison of models across different scenarios in the causal discovery task.

A.3.6. Transfer Study

In the following we show the performance of different models on the training set, a test set of the same distribution, and a set drawn from an unseen environment for different scenarios. As in Section 5.5, we use the L2-Loss on samples of an unseen environment to measure out-of-distribution generalization. Figure A.5, Figure A.6 and Figure A.7 show results according to the underlying mechanisms, target variable or type of intervention respectively. The boxes show the quartiles and the upper whiskers ranges from third quartile to $1.5 \cdot IQR$ where IQR is the interquartile range. Similar for the lower whisker.

A.4. Experimental Details Colored MNIST

For the training, we use a feed forward neural network consisting of a feature selector followed by a classifier. The feature selector consists of two convolutional layers with a kernel size of 3 with 16 respectively 32 channels followed by a max pooling layer with kernel size 2, one dropout layer ($p = 0.2$) and a fully connected layer mapping to 16 feature dimensions. After the first convolutional layer and after the pooling layer a PReLU activation function is applied. For the classification we use a PReLU activation function followed by a Dropout layer ($p = 0.2$) and a linear layer which maps the 16 features onto the two classes corresponding to the labels.

We use the data generating process from [34]. 50 000 samples are used for training and 10 000 samples as test set. For training, we choose a batch size of 1000 and train our models for 60 epochs. We choose a starting learning rate of $6 \cdot 10^{-3}$. The learning rate is decayed by 0.33 after 20 epochs. We use an L2-Regularization loss weighted by 10^{-5}. After each epoch, we randomly reassign the colors and the labels with the corresponding probabilities. The one-dimensional Wasserstein loss is applied dimension-wise and the maximum over dimensions is computed in order to compare residuals. For the HSIC we use a cauchy kernel with $\sigma = 1$. The invariance loss $\mathcal{L}_I$ is simply the sum of the HSIC and Wasserstein term. For Figure 5.7 we trained our model with $\lambda_I \approx 13$. This hyperparameter is chosen from the best run in Figure 5.8. For stability in the case of large λ_I, we divide the total loss by λ_I during training to produce the results in Figure 5.8. For the reported accuracy of IRM, we train with the same network architecture on the dataset where we created training instances online.

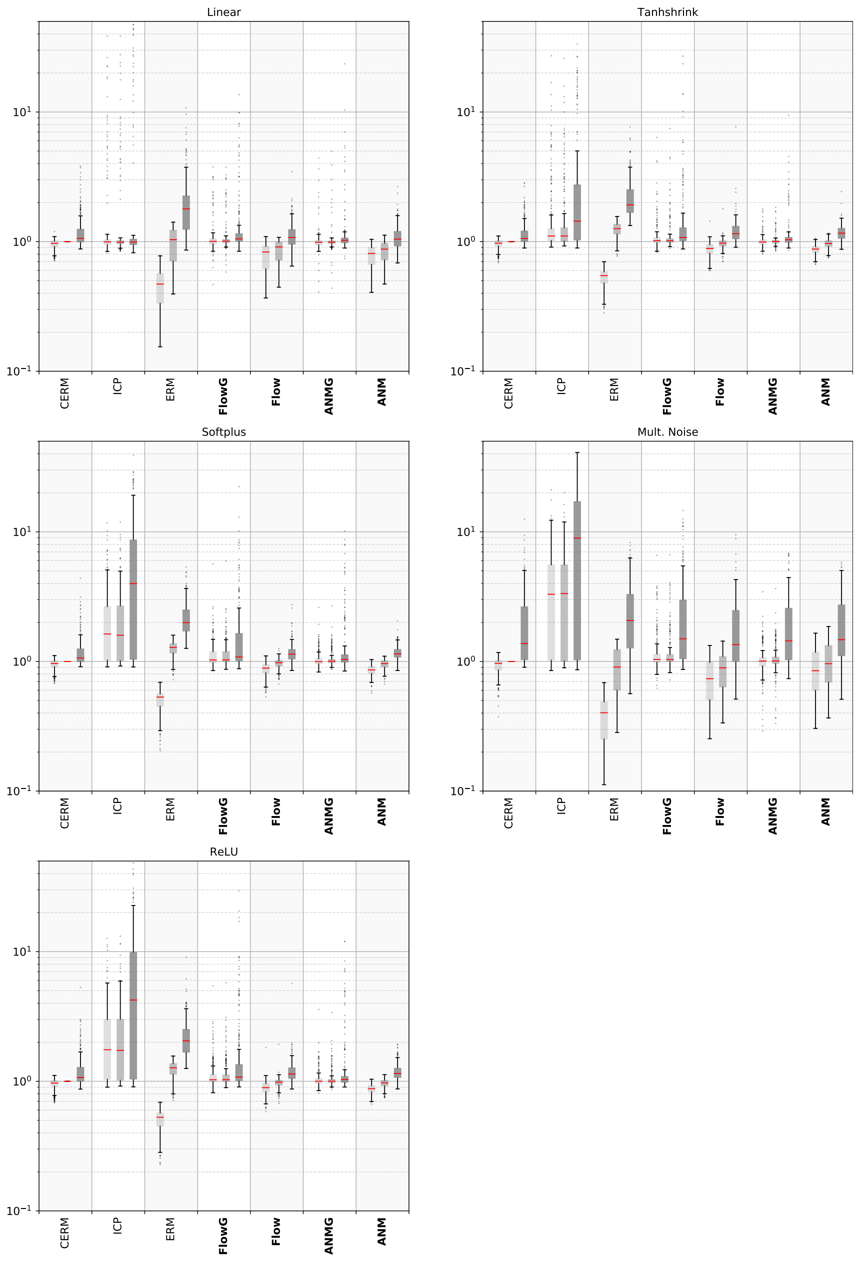


Figure A.5. Logarithmic plot of L2 errors, normalized by CERM test error. For each method (ours in bold) from left to right: training error, test error on seen environments, domain generalization error on unseen environments. Scenarios for different mechanisms are shown.

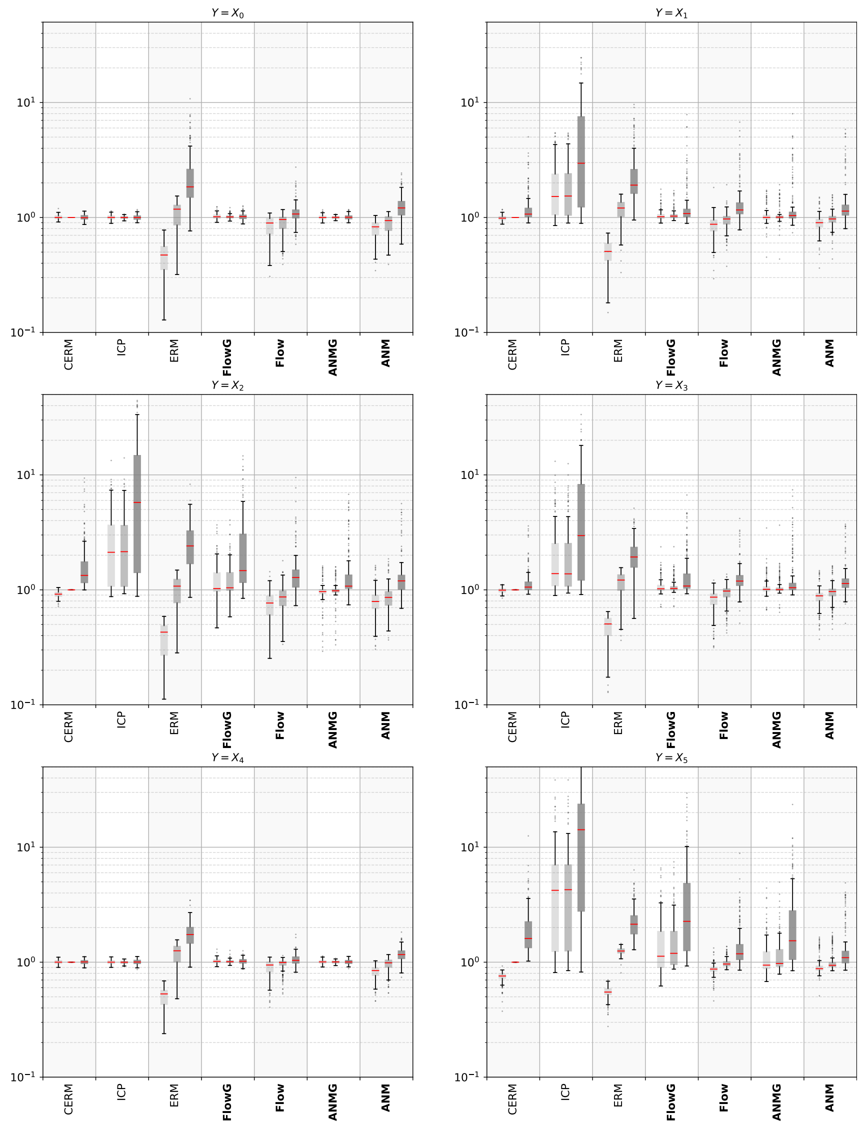


Figure A.6. Logarithmic plot of L2 errors, normalized by CERM test error. For each method (ours in bold) from left to right: training error, test error on seen environments, domain generalization error on unseen environments. Scenarios for different target variables are shown.

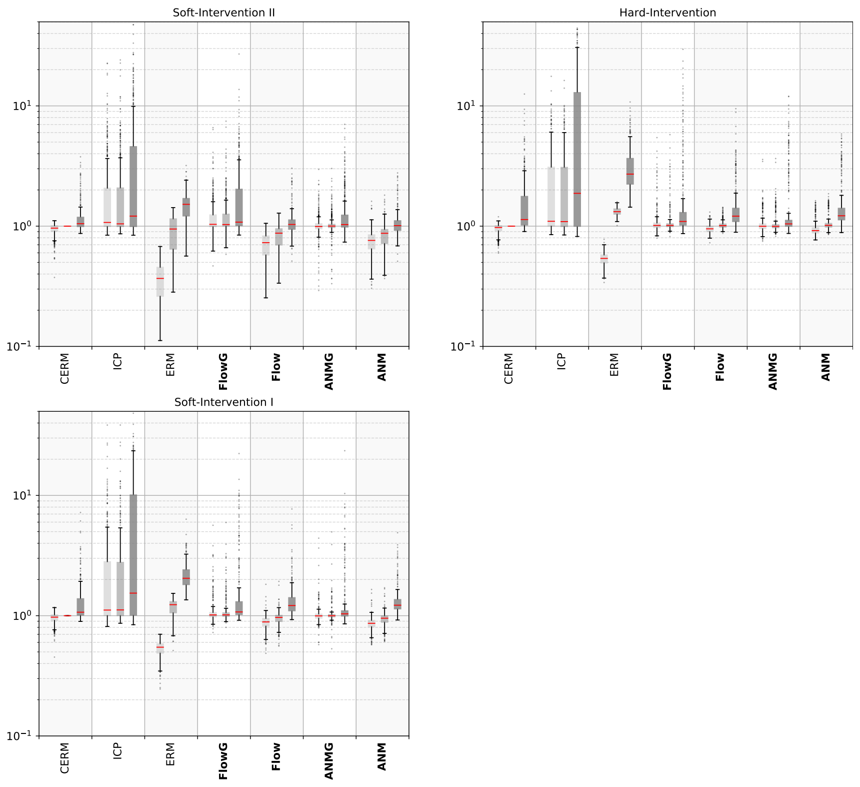


Figure A.7. Logarithmic plot of L2 errors, normalized by CERM test error. For each method (ours in bold) from left to right: training error, test error on seen environments, domain generalization error on unseen environments. Scenarios for different intervention types are shown.

B Appendix: Context-Aware DG

B.1. Theory

In the following, we discuss the assumptions in (c) and (d). In our experiments, we observed that in most datasets a relatively small sample size suffices to infer the environment label with approximately 100% accuracy (see Table B.2). Therefore, the assumption that there exists a function $g(\mathbf{S}^{(n)}) = E$ seems justified if n is sufficiently large. To generalize the assumption where the environment label is not fully inferable, we have to make assumptions. For one, we require $\mathbf{S}^{(n)} \mid Z \mid \mathbf{X}$. This can be interpreted as "increasing the set size does not improve the prediction of E" in a contextual environment model. Also $\mathbf{S}^{(n)} \perp Z \mid \mathbf{X}, Y$ can be interpreted similarly: increasing the set size and considering the ground truth label/value does not enhance the predictability of E. Both assumptions should hold approximately if n is large enough. With the assumption $I(Y; E \mid \mathbf{X}) > I(Z; \perp Y \mid \mathbf{X})$ we assume that the noise Z is less predictive of Y compared to E if $\mathbf{X}$ is given. This can be roughly interpreted as the noise does not prove useful for predicting Y from $\mathbf{X}$ compared to the ground truth environment label.

B.2. Experiments: General Remarks

Due to the large amount of settings, we did only little hyper-parameter optimization (we looked into batch size, learning rate, and network size). For a given dataset we optimized only on one scenario where an environment is left out during training. The found hyper-parameters were then applied on all other scenarios. To ensure that the baseline model is comparable to ours, we ensure that the inference network (and feature extractor) in Figure 6.5 have a comparable number of parameters as the baseline model. In all cases, the set-encoder is kept simple and its hyper-parameters are selected for optimal performance of the contextual environment predictor $f^{E|\mathbf{X},\mathbf{S}^{(n)}}$. For an overview, see Table B.2. Throughout all experiments, we employ a mean-pooling operation.

We show the accuracies of classifying the environment of the contextual-environment model $f^{E|\mathbf{X},\mathbf{S}^{(n)}}$ and the baseline environment model $f^{E|\mathbf{X}}$ in Table B.2. Here we only consider the datasets where we performed a full evaluation of all criteria.

B.3. Experiment 1: Details

B.3.1. Data Generation

Simpson's Paradox [32, 122] describes a statistical phenomenon wherein several groups of data exhibit a trend, which reverses when the groups are combined. There are several famous real-world examples of Simpson's Paradox, such as a study examining a gender bias in the admission process of UC Berkeley [283] or an evaluation of the efficacy of different treatments for kidney stones [284].

In order to replicate this, we create a dataset as a mixture of 2D multivariate normal distributions, with the intent of using the first dimension as a feature, and the second as a regression target. Unless otherwise specified, we generate the data by taking an equal number of samples from each mixture component, defining the environment as a one-hot vector over the mixture components.

The mixture components are chosen to lie on a trend line that is opposite to the trend within each mixture. We achieve this by using a negative global trend, and choosing the covariance matrix of each mixture as a scaled and rotated identity matrix with opposite trend.

Setting	**Value**	**Controls**
`n_domains`	5	number of mixture components
`n_samples`	10000	number of samples per mixture component
`spacing`	2.0	spacing between means of the mixture components
`noise`	0.25	overall noise level
`noise_ratio`	6.0	ratio of the primary to secondary noise axis
`rotation_range`	$(45.0, 45.0)$	min (leftmost) and max (rightmost) mixture rotation angle

Table B.1. Default Settings for the Simpson's Paradox Dataset. Samples from the dataset constructed with these settings can be seen in Figure 6.6

B.3.2. Training Details

We consider five distinct settings, where in each setting, one domain is left out during training, and considered for evaluation as a novel environment. To gauge the uncertainty stemming from data sampling, we also consider five dataset seeds for partitioning into training, validation, and test sets. For each dataset seed and model, we consider the results due to the best performance on the validation set.

We enforced that our approach and the baseline model have a similar amount of parameters for the feature extractor and final inference model. We conducted minimal hyperparameter tuning (focusing on parameters such as the learning rate schedule, batch size, and the number of parameters), and this was performed solely within one "leave-one-environment-out" setting. In total, we trained the five models outlined in Table 6.1 using five distinct dataset seeds. Consequently, a total of $5 \cdot 5 \cdot 5 = 125$ models were trained.

In all cases, the set-encoder is kept simple and its hyper-parameters are selected for optimal performance of the contextual environment predictor $f^{E|\mathbf{X},\mathbf{S}^{(n)}}$. We choose the mean as the pooling operation.

B.3.3. Non-Linear Models

In the experiments in Subsection 6.8.2, we considered linear models for our model and the baseline. In the following, we show results for the non-linear model class in Figure B.1. We compare predictions of a baseline model and our model on all environments in Figure B.2. We see that the extrapolation task fails in some cases as in environment 1. This is due to the mismatch of the considered model class and ground truth model.

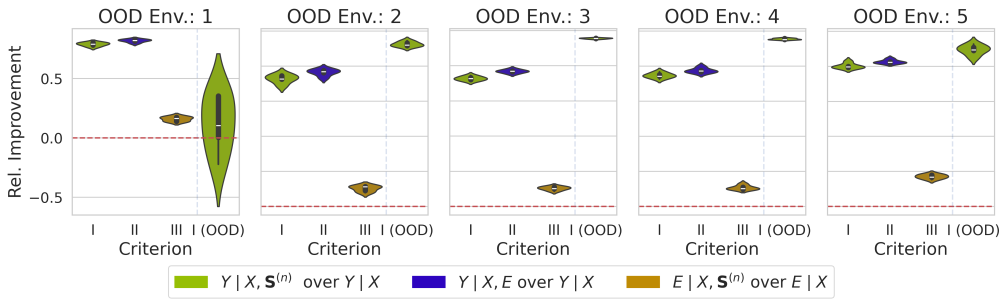


Figure B.1. Experiment 1. Verification of criteria. In I we depict the relative improvement of our approach versus a baseline model. We also show I (OOD) on OOD data. In II we show the relative improvement of the oracle model compared to the baseline. In III we compare the relative improvement of the contextual environment model with respect to the baseline environment model.

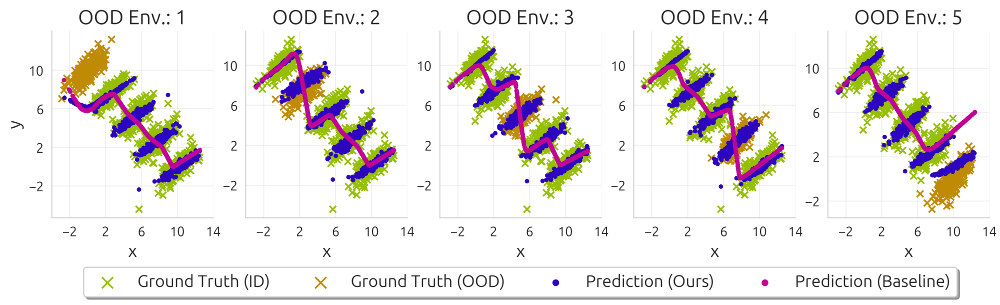
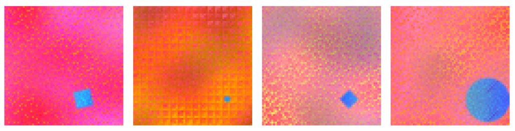


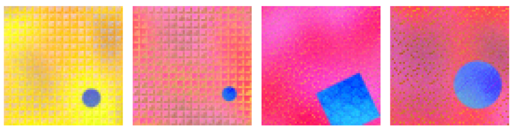
Figure B.2. Experiment 1. Models are trained on all environments except the OOD environment. "Extrapolation", i.e. when environment 1 or 5 is OOD, is a particularly hard task in this setting. The set-based model shows slightly better extrapolation capabilities. Generally, our model exhibits adaptability to diverse environments, addressing a limitation present in the baseline model.

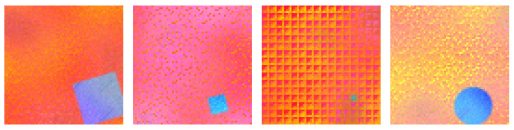
B.4. Experiment 2: Details

Data samples from different environments are depicted in Figure B.3. The process of how inputs relate to outputs is described in Subsection 6.8.3.

During training, we employ a convolutional network to extract features $g(\mathbf{X})$. These features are passed to the inference network and the set-encoder. The feature extractor is


(a) Environment 1


(b) Environment 2


(c) Environment 3

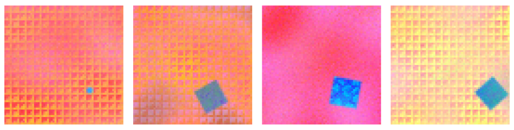

(d) Environment 4

Figure B.3. Experiment 2. We generate four distinct domains synthetically. Notably, the background color within each domain follows a normal distribution. However, there are variations in the means across these domains Note that there is a huge overlap between the environments.

then jointly trained with the inference network and set-encoder. We ensured that the feature extractor plus inference network and the baseline model have a comparable amount of parameters. The set-encoder is kept simple and its hyper-parameters are selected for optimal performance of the contextual environment predictor $f^{E|\mathbf{X},\mathbf{S}^{(n)}}$. As a pooling operation we choose the mean-pooling.

B.5. Experiment 3: Details

To select between the baseline model and the invariant model, we are required to distinguish between ID and OOD data. Therefore, we follow the approach proposed in Subsection 6.6.7. We consider the k-nearest neighbors of the training set to compute the score s_ψ where $k = 5$. Since we compare the scores elicited by features of the baseline model with the scores elicited by the features extracted by the set-encoder, we restricted both architectures to have the same feature dimension. To establish a threshold for distinguishing between ID and OOD samples, we designate samples with scores below the 95% quantile of the validation set as ID and those above as OOD (see Subsection 6.6.7 for details).

In total, we explore five dataset seeds to partition into training, validation, and test sets. To train an invariant model, we considered the same split in training, validation, and test set where the background color has no association with the label. Therefore the invariant model learns to ignore the background color and only utilize the shape for prediction. To learn effectively about the environment, we considered a large set input, namely 1024 samples in $\mathbf{S}^{(n)}$. We employed a simple set-encoder incorporating a mean pooling operation.

B.6. Experiment 4 and 5: Details

For the BikeSharing dataset we consider a simple feed-forward neural network in all models. For the PACS as well as the OfficeHome dataset we consider features $g(\mathbf{X})$ that are kept fixed and not optimized. Here, we employ the Clip features proposed in [63]. The inference model, baseline model, and set-encoder are kept simple and employ only linear layers followed by ReLU activation functions. Given that Clip features considerably simplify the task, we performed a minimal hyper-parameter search and ensured that the inference model had a similar number of parameters as the baseline model. In all cases, the set-encoder is kept simple and its hyper-parameters are selected for optimal performance of the contextual environment predictor $f^{E|\mathbf{x},\mathbf{s}^{(n)}}$.

Dataset / Set Size	Simpson / 32				
Domain	1	2	3	4	5
$f^{E\|\mathbf{x}}$	86.3 ± 1.3	90.8 ± 1.3	90.7 ± 0.8	90.4 ± 0.9	85.5 ± 0.8
$f^{E\|\mathbf{x},\mathbf{s}^{(n)}}$	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0	**100.0** ± 0.0

Dataset / Set Size	ProDAS / 128				OfficeHome / 4	PACS / 4
Domain	1	2	3	4	Product	Art
$f^{E\|\mathbf{x}}$	43.8 ± 1.1	50.0 ± 1.3	49.9 ± 2.3	44.4 ± 1.0	86.16 ± 0.33	**99.72** ± 0.33
$f^{E\|\mathbf{x},\mathbf{s}^{(n)}}$	**99.6** ± 0.6	**99.5** ± 1.0	**98.7** ± 1.6	**98.0** ± 3.2	**98.49** ± 0.24	**100.0** ± 0.0

Table B.2. Environment classification accuracy for different models and datasets, broken down by domain. As in Table 6.3, the uncertainty (mean and standard deviation) is computed over multiple seeds for dataset splits. In all cases, the set-based model outperforms the baseline.

In all cases, the set-encoder is kept simple and its hyper-parameters are selected for optimal performance of the contextual environment predictor $f^{E|\mathbf{x},\mathbf{s}^{(n)}}$.

B.7. Comparison of Permutation-Invariant Architectures

As a pilot experiment, we estimate the contextual information contained in a set input by evaluating the binary classification accuracy of a set-based model compared to a baseline model with singleton sample input.

Importantly, we postulate that for stronger domain overlap, the contextual information contained within the single sample decreases significantly, while the contextual information within the set decreases only weakly, depending on the set size. Domains that do not overlap exactly will remain distinguishable, so long as the set size is large enough.

Therefore, we construct the toy dataset as described in Appendix B.3.1, but use the setting `n_domains = 2` and vary the distance between environments for each experiment.

We train each architecture on this dataset for 20 epochs, using 5 different seeds. We evaluate a total of 30 domain spacings, linearly distributed between 0.05 and 1.5 (both inclusive). Since we evaluate a baseline model, plus 3 set-based models at 3 different set sizes, this brings us to a total of $30 \cdot 20 \cdot 5 \cdot (1 + 3 \cdot 3) = 30000$ model epochs. We choose the batch size at 128 fixed.

Each architecture consists of a linear projection into a 64-dimensional feature space, followed by a fully connected network with 3 hidden layers, each containing 64 neurons and a ReLU [285] activation. For the set-based methods, this is followed by the respective pooling. We choose 8 heads for the attention-based model.

Finally, the output is linearly projected back into the 2-dimensional logit space, where the loss is computed via cross-entropy [286].

For methods that support a non-unit output set size, we choose the output set size as 4. The output set is mean-pooled prior to projection into the logit space. Results are shown in Figure B.4.

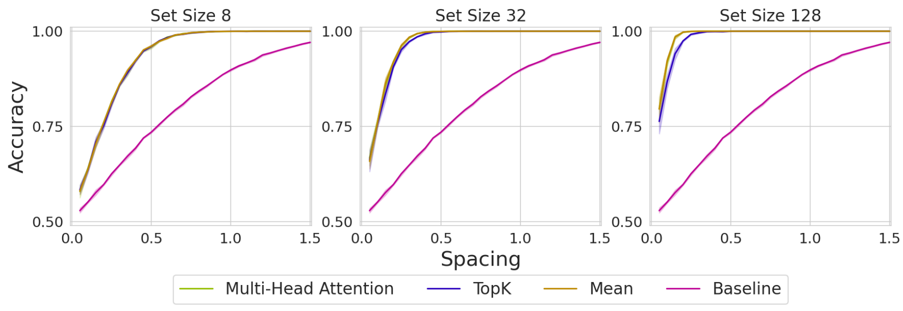


Figure B.4. Comparison of different architectural choices for the permutation-invariant network in predicting the data's originating environment. We consider various distances between environments and different set sizes n. As anticipated, the plots illustrate that smaller environment distances make it more challenging to differentiate between them. Moreover, with larger set size n, our ability to predict the environment label improves. Notably, the baseline model shows significantly poorer performance compared to the model utilizing contextual information in the form of a set input.

B.8. Bike Sharing Dataset

This dataset, taken from the UCI machine learning repository[247], consists of over 17000 hourly and daily counts of bike rentals between 2011 and 2012 within the Capital bike-share system.

Each dataset entry contains information about the season, time, and weather at the time of rental. Casual renters are also distinguished from registered ones.

Similar to [287], we only consider the hourly rental data. We drop information about the concrete date and information about casual versus registered renters. We choose the season variable (spring, summer, fall, winter) as the environment and the bike rental count as the regression target. Since we deal with count data, we also apply square root transformation on the target similar to [287].

C Appendix: Competence Regions in DG

In the following, we describe optimization procedures in detail, give additional detailed results and describe the open world datasets in detail for Chapter 7.

C.1. Detailed Qualitative Results

Figure C.1 and Figure C.2 show for the PACS dataset the three images attaining the highest and the lowest incompetence scores per class respectively. Images with lower scores achieve higher accuracy compared to the highest-scored images.

C.2. Detailed Quantitative Results - Dependence on Competence Threshold

We show the accuracy on OOD test data in dependence on the competence threshold α for the DG datasets PACS, OfficeHome, VLCS, TerraIncognita, DomainNet and SVIRO in Figure C.3, Figure C.4, Figure C.5, Figure C.6, Figure C.7 and Figure C.8 respectively. We only show results for Deep-KNN, Logit, ViM and GMM applied on the ERM classifier. For Deep-KNN, Logit, and Vim we see in almost all cases the monotonic behavior as predicted in Proposition 8. On the DG datasets VLCS, DomainNet and TerraIncognita the GMM score fails to show this monotonic behavior for some test domains. Therefore, GMM has not the monotonic behavior we would expect from an admissible incompetence score. For some domains, all scores do not behave as we aimed for. For instance, in the LabelMe test domain in VLCS (see Figure C.5) we cannot achieve the ID accuracy for all thresholds α and all incompetence scores. While this behavior is extremely rare in our experiments (for the feature- and logit-based scores), it shows that the current competence scores can fail for some domain shifts.

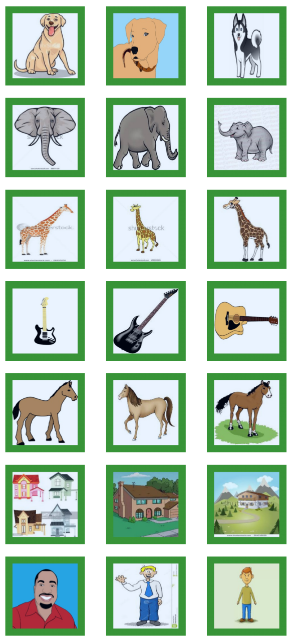
In Figure C.9 we also show results for different thresholds according to their percentile in the ID distribution. We can see that the relative performance of the scores stays considerably stable.

C.3. Detailed Quantitatively Results – Extensive Study

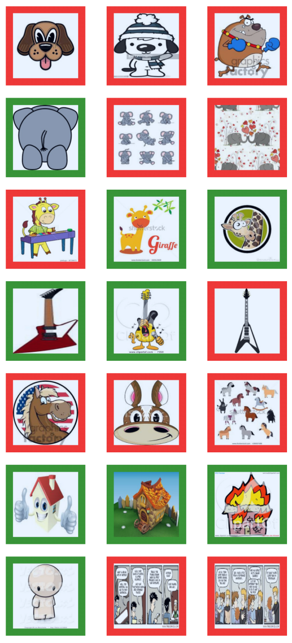
In Table C.1, Table C.2 and Table C.3 we give detailed results for all DG datasets considered in this work: PACS, VLCS, OfficeHome, TerraIncognita, DomainNet, and SVIRO. We list the accuracies in the competence region where the incompetence threshold is chosen as the

(a) Lowest scores Art.

(b) Highest scores Art.



(c) Lowest scores Cartoon.

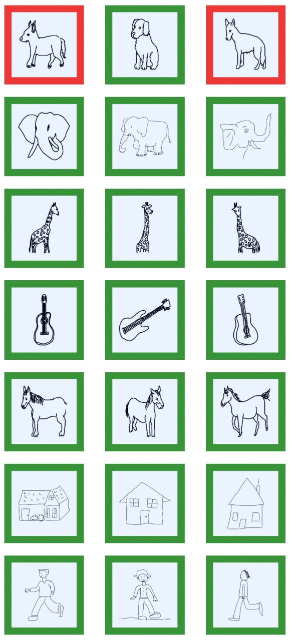


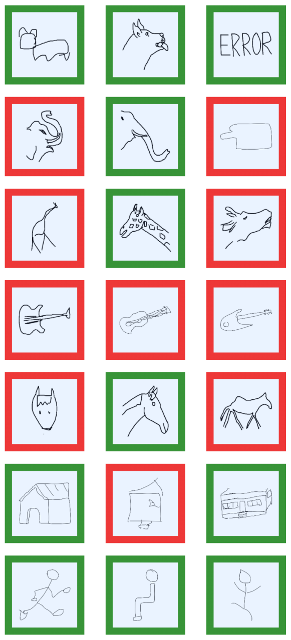
(d) Highest scores Cartoon.

Figure C.1. Images with highest and lowest scores for different domains on PACS. Scores are computed with Deep-KNN on ERM.

(a) Lowest scores Photography.

(b) Highest scores Photography.


(c) Lowest scores Sketch.



(d) Highest scores Sketch.

Figure C.2. Images with highest and lowest scores for different domains on PACS. Scores are computed with Deep-KNN on ERM.

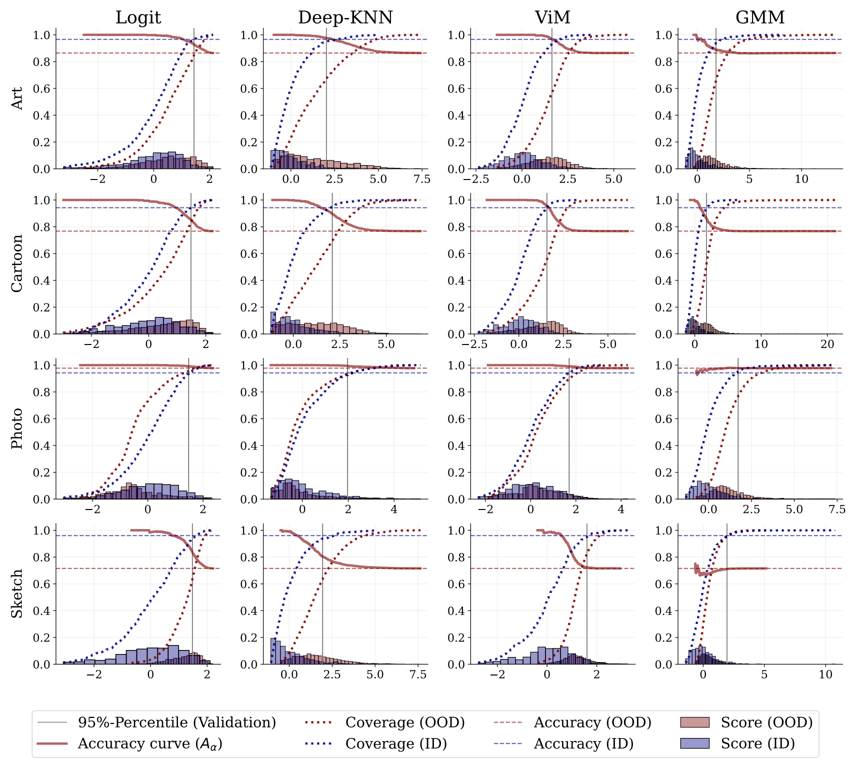


Figure C.3. The accuracy of the ERM classifier on OOD data $A_{\text{OOD}}(\alpha)$ as the competence region is enlarged by increasing the allowed incompetence α. Here we show the results for all DG tasks of the PACS dataset.

95% percentile of the ID validation set. Here we show the median, the 5% and 95% percentiles over all test domains and classifiers. We can see that the deviations between different test domains are quite severe indicating different strengths of domain shifts across the DG tasks. The main observations in Subsection 7.4.4 (e.g. the OOD-gain is quite significant and feature-based methods [ViM; Deep-KNN] are very successfull) hold across the different DG tasks.

C.4. Open World Setting

Open World Creation We use additional data to extend the closed world datasets to the open world setting. We use similar domains of other datasets with disjunct classes to generalize the DG datasets. The ID datasets and the open world extensions are listed in Table C.4. We show examples of test data (closed world) and open world samples for all DG datasets. For PACS (in Figure C.14) for VLCS (in Figure C.15), for OfficeHome (in Figure C.16) and for TerraIncognita (in Figure C.17)

Open World Results Figure C.10 shows the ID-Gain and OOD-Gain for all incompetence scores considered in this work depending on the fraction of open world samples. We see that ViM and Deep-KNN are particularly able to delineate unknown class instances from known class instances resulting in an improved ID- and OOD-Gain across all open world fractions.

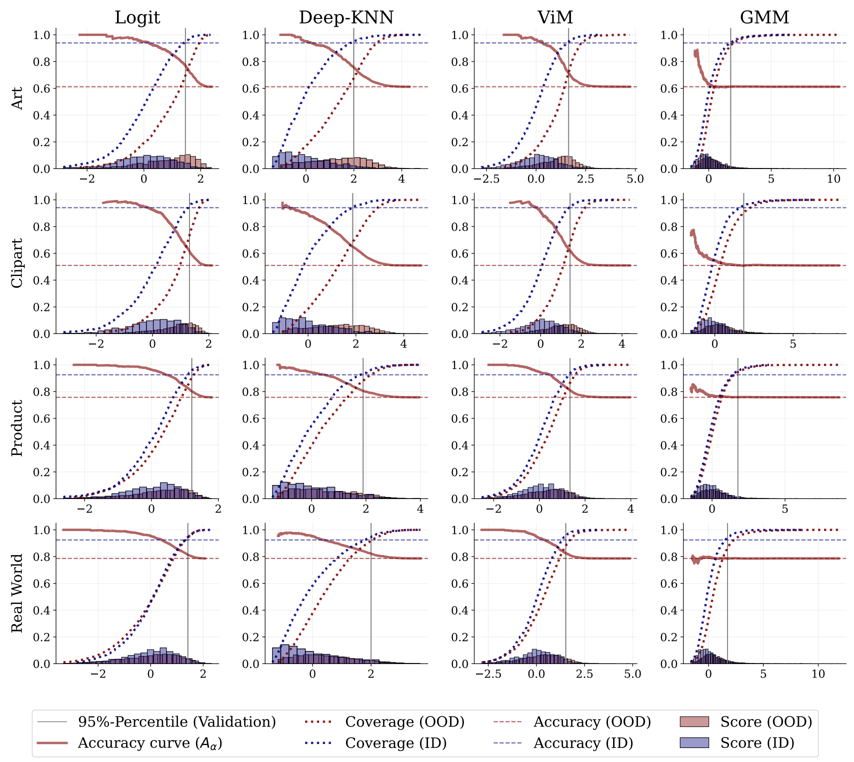


Figure C.4. The accuracy of the ERM classifier on OOD data $A_{OOD}(\alpha)$ as the competence region is enlarged by increasing the allowed incompetence α. Here we show the results for all DG tasks of the OfficeHome dataset.

In Figure C.11 we investigate the behavior of different scores in detail. It shows the AUROC of delineating ID data vs. correctly classified samples of the test domain, ID data vs. wrongly classified samples of the test domain, and ID data vs. unknown class instances in general. Here we consider an unknown test domain where 25% of all samples are open world outliers. We see an interesting behavior here: ViM and Deep-KNN are well-able to filter out wrongly classified samples, but also filter out many correctly classified samples. The logit-based scores (Logit, Softmax, Energy, Energy-React) are less successful in filtering out wrongly classified samples, but also keep more correctly classified samples. In the optimal case, we would expect that the AUROC of ID vs. correct test data is ≤ 0.5 and the AUROC of ID vs. false OOD data is 1. This would imply that we could successfully filter out wrongly predicted samples and keep a high coverage. Figure C.11 shows that ViM and Deep-KNN are capable of filtering out new class instances across all DG datasets. For all other scores, we can find datasets where this behavior is not achieved. Consequently, ViM and Deep-KNN work best when unknown class instances occur.

C.5. Training Details and Classifiers

All classifiers are trained using the DomainBed repository [1]. We train three different neural network architectures with Emprirical-Risk-Minimization, shortly ERM [64]. Namely, a ResNet based architecture [276], a Vision Transformer [277] and a Swin Transformer

[1]`https://github.com/facebookresearch/DomainBed`

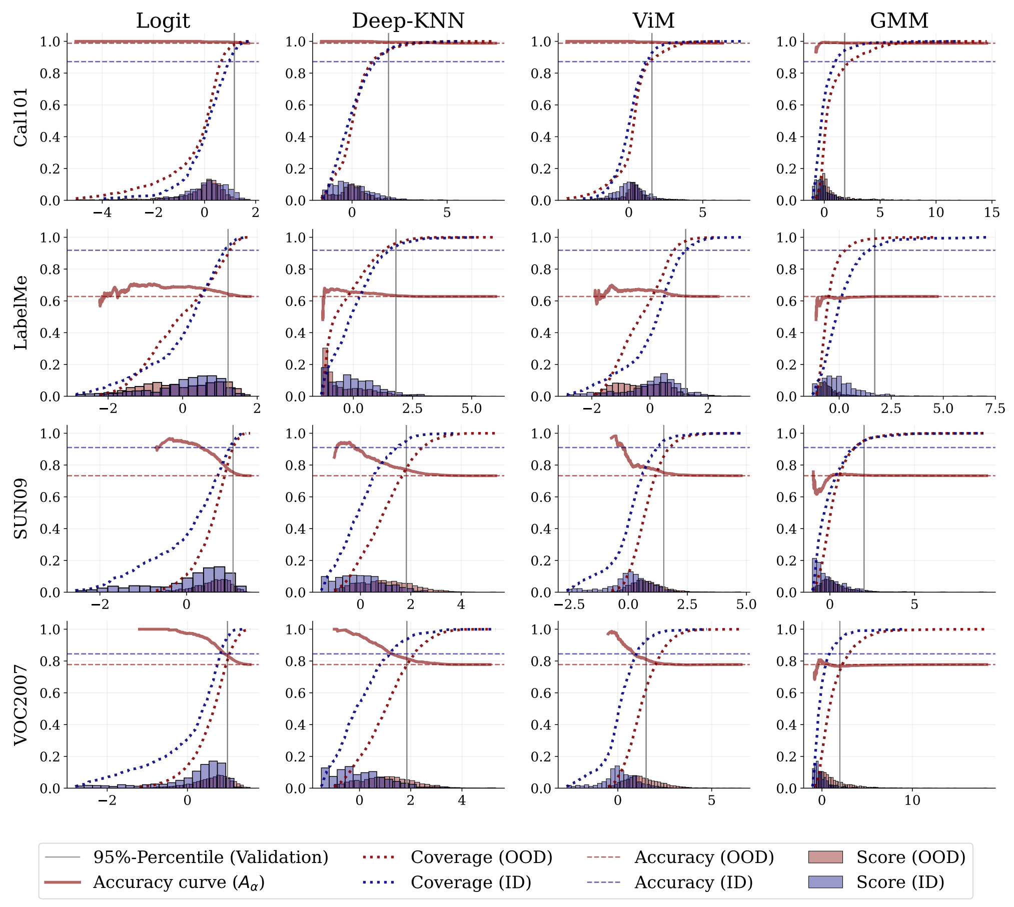


Figure C.5. The accuracy of the ERM classifier on OOD data $A_{OOD}(\alpha)$ as the competence region is enlarged by increasing the allowed incompetence α. Here we show the results for all DG tasks of the VLCS dataset.

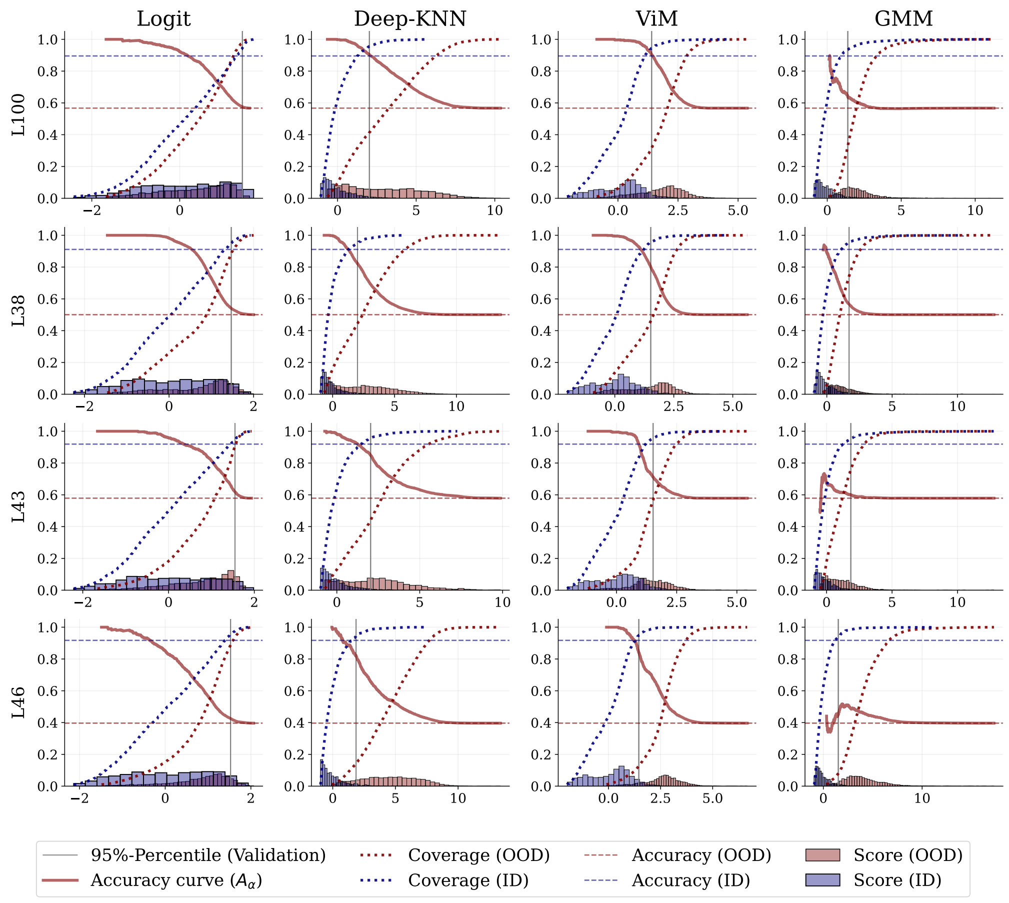


Figure C.6. The accuracy of the ERM classifier on OOD data $A_{OOD}(\alpha)$ as the competence region is enlarged by increasing the allowed incompetence α. Here we show the results for all DG tasks of the TerraIncognita dataset.

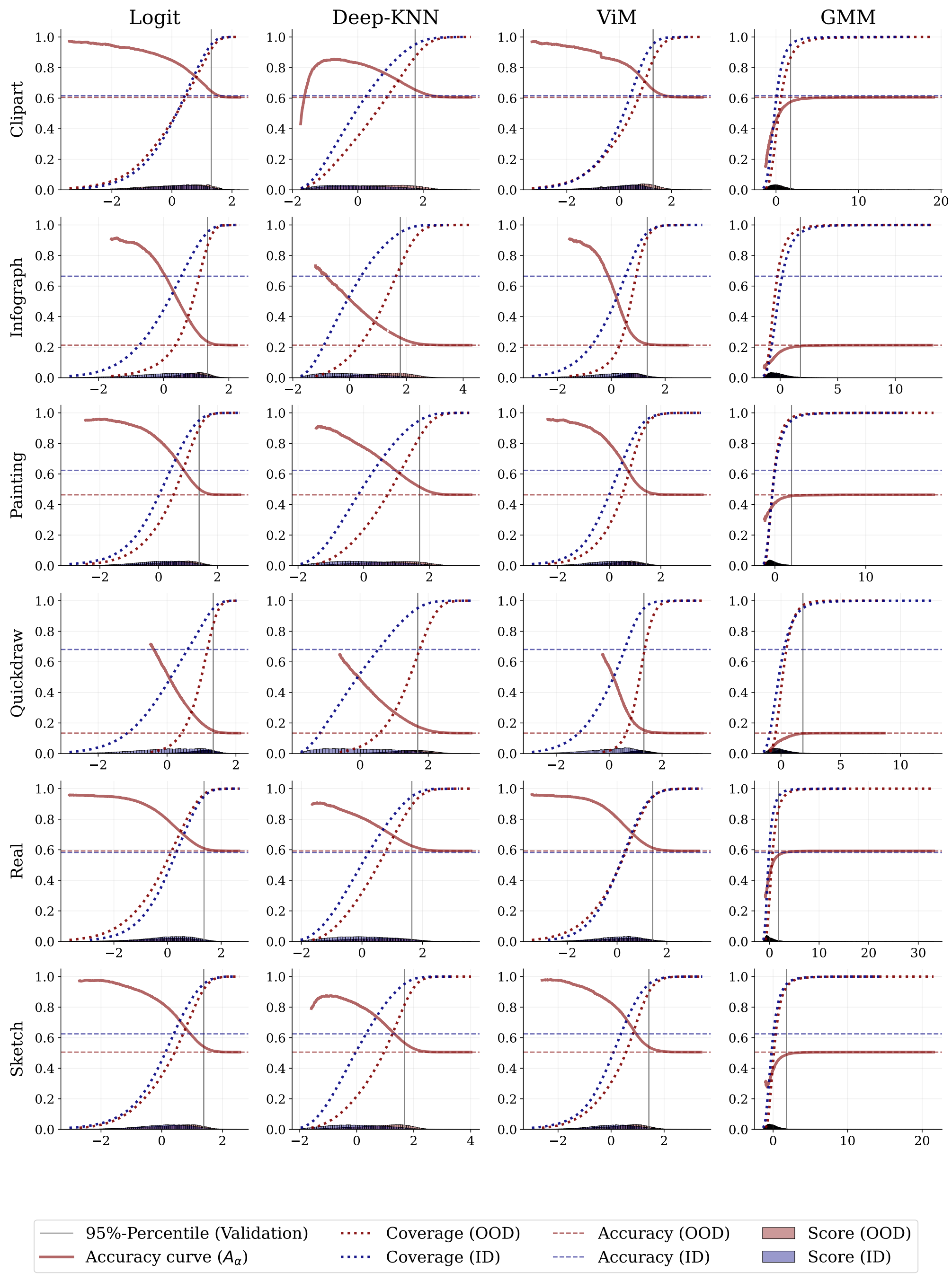


Figure C.7. The accuracy of the ERM classifier on OOD data $A_{\text{OOD}}(\alpha)$ as the competence region is enlarged by increasing the allowed incompetence α. Here we show the results for all DG tasks of the DomainNet dataset.

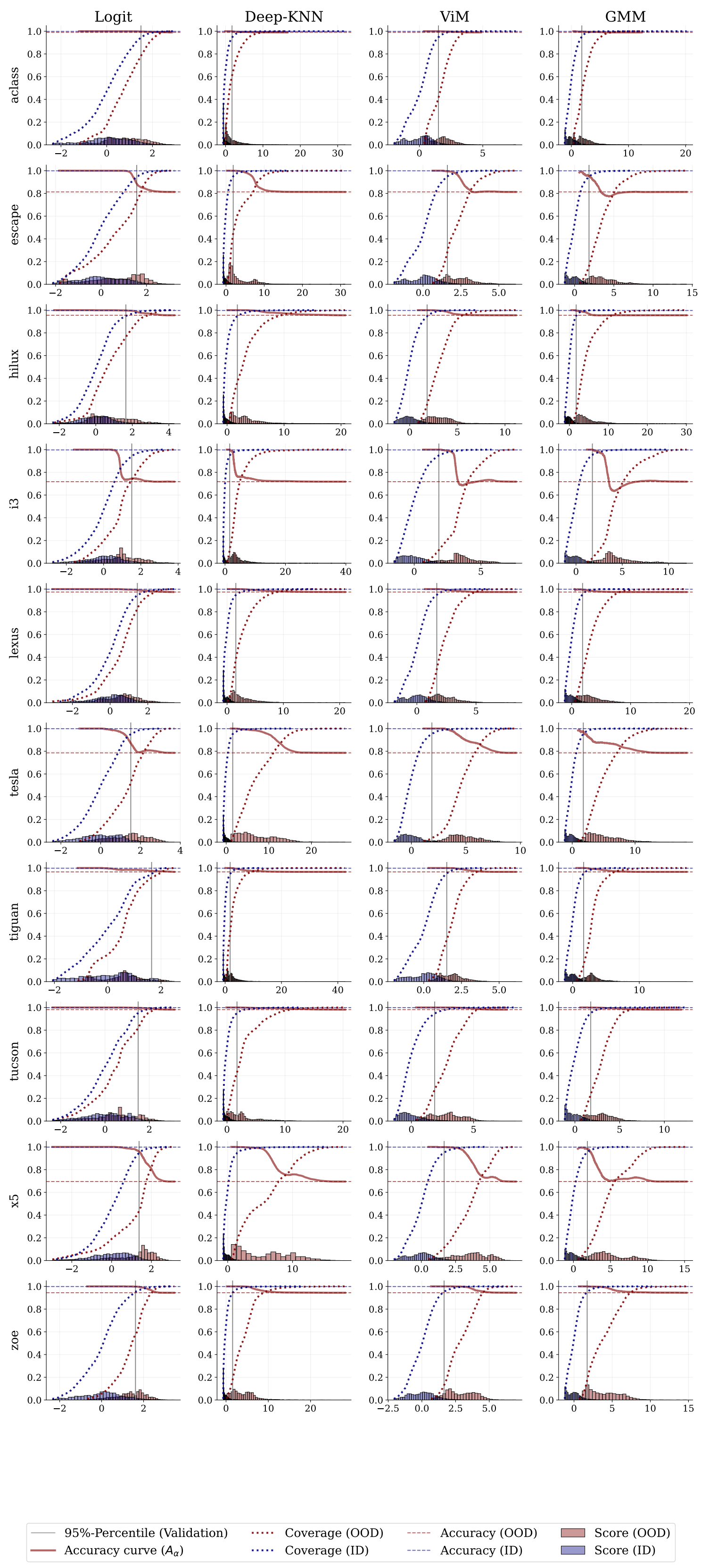


Figure C.8. The accuracy of the ERM classifier on OOD data $A_{OOD}(\alpha)$ as the competence region is enlarged by increasing the allowed incompetence α. Here we show the results for all DG tasks of the SVIRO dataset.

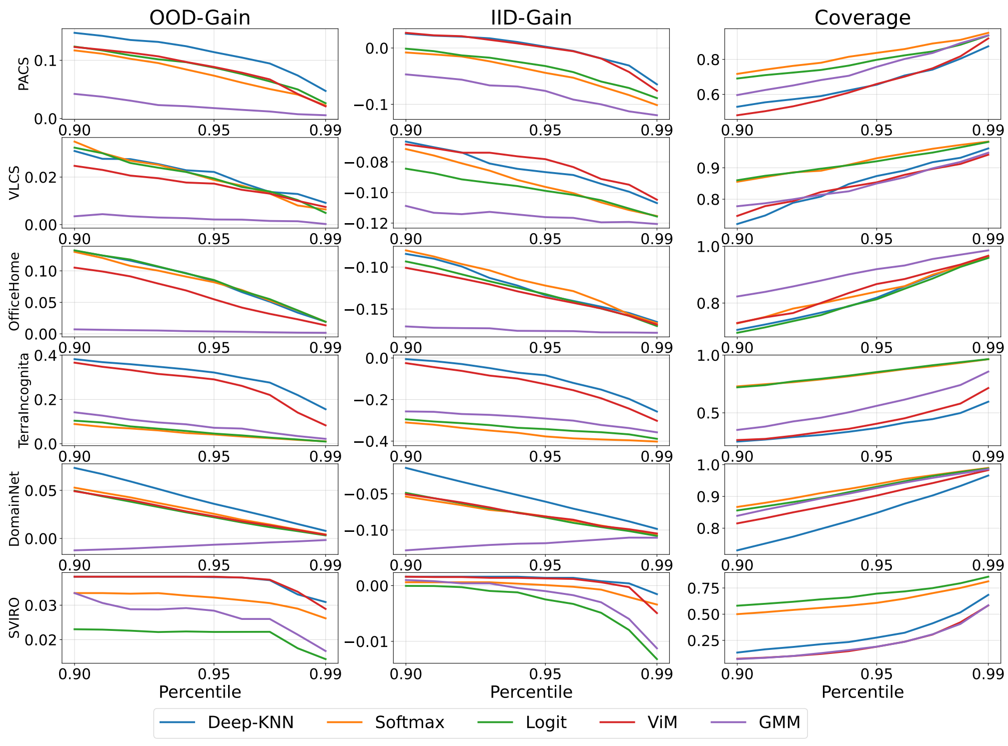


Figure C.9. Median of accuracies in competence region for different thresholds (percentiles of ID distribution) over all domain roles and classifiers.

[278]. If we just refer to ERM, we mean the ResNet-based architecture. Furthermore, we train classifiers with various recent DG algorithms, namely Fish [262], GroupDRO [273], SD [274], SagNet [65], Mixup [275] and VREx [126].

We use all the standard settings provided in the DomainBed repository and train all classifiers with hyperparameters proposed in the repository. The Vision Transformer and SwinTransformer are trained with hyperparameters found useful on these datasets and architectures as in [288]. Each model is trained for 100 epochs on the smaller datasets (PACS, VLCS, TerraIncognita and OfficeHome) and for 10 epochs on DomainNet and SVIRO. When no improvement in terms of accuracy on the validation set is achieved, we stop the training. The best model is chosen due to the accuracy on the ID distribution measured via the accuracy on the validation set.

Some scores are computed on the logits and some on the features. If computed on the features, we use the output of the penultimate layer of the model as input to the score function. We distinguish between training, validation, and test sets of the ID distribution. For the OOD distribution we only consider one dataset provided by the DG task which is not seen during training. Score quantiles are always computed on the ID validation set. The ID accuracy is computed on the ID test dataset. If score functions need optimization (as with GMM), we train them on the ID training set. If a score function needs optimization, we restrict the training set to 50 000 samples. This only affects the DomainNet dataset. We do only little to no optimization of the parameters of the score functions. We mainly stay in line with the standard settings found in the literature. For Deep-KNN we choose $K = 1$ since it shows slightly improved performance on the ID distribution (only inspected on PACS).

	In Percentages (%)	Art			Cartoon			Painting			Sketch		
		OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑
PACS	Deep-KNN	**12** [8-14]	**2** [-1-3]	66 [57-72]	14 [11-18]	-1 [-6-1]	63 [56-71]	**2** [1-2]	**3** [3-5]	94 [93-96]	**13** [9-17]	-9 [-15–5]	59 [53-70]
	ViM	11 [7-14]	1 [-2-3]	56 [53-71]	**17** [11-19]	**0** [-4-2]	55 [49-60]	**2** [1-2]	**3** [3-5]	91 [86-95]	3 [1-21]	-15 [-22–5]	77 [35-86]
	Softmax	7 [6-10]	-4 [-5-1]	86 [77-88]	7 [5-10]	-10 [-12–4]	82 [78-86]	1 [1-2]	**3** [2-5]	97 [96-98]	12 [7-15]	**-8** [-19–5]	68 [63-82]
	Logit	9 [6-11]	-3 [-4-1]	82 [65-85]	10 [7-11]	-9 [-10–2]	77 [72-84]	**2** [1-2]	**3** [3-5]	95 [93-97]	12 [4-16]	**-8** [-22–5]	65 [57-90]
	Energy	9 [6-11]	-3 [-4-1]	81 [64-84]	9 [7-11]	-9 [-10–2]	77 [72-84]	**2** [1-2]	**3** [3-5]	95 [92-97]	11 [3-16]	**-8** [-23–4]	66 [56-91]
	Energy-React	9 [6-11]	-3 [-4-1]	80 [63-84]	9 [7-11]	-8 [-10–2]	77 [71-84]	**2** [1-2]	**3** [3-5]	95 [92-97]	11 [3-16]	**-8** [-23–4]	66 [56-91]
	Mahalonobis	2 [0-10]	-7 [-11–1]	67 [56-94]	7 [0-12]	-10 [-14–5]	62 [50-94]	0 [0-1]	**3** [2-4]	87 [78-96]	0 [-1-17]	-19 [-24–9]	89 [28-97]
	GMM	3 [1-10]	-7 [-11-2]	66 [54-86]	8 [4-13]	-9 [-15–2]	56 [50-67]	1 [0-1]	**3** [2-4]	83 [76-94]	0 [-1-17]	-19 [-24–9]	87 [28-97]
	PCA	0 [-1-7]	-11 [-14-0]	79 [63-93]	3 [1-11]	-13 [-17–4]	72 [56-81]	0 [-1-1]	**3** [0-4]	82 [70-94]	0 [-1-13]	-18 [-24–12]	95 [28-100]
		Cal101			Label			SUN09			VOC2007		
		OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑
VLCS	Deep-KNN	**2** [1-4]	13 [11-15]	93 [89-96]	0 [0-1]	-27 [-28–22]	98 [96-99]	2 [1-5]	**-15** [-19–6]	79 [73-85]	**4** [4-6]	**-6** [-9–2]	76 [66-81]
	ViM	**2** [0-3]	**14** [10-15]	87 [68-88]	0 [0-0]	-27 [-28–22]	98 [97-99]	2 [1-4]	-16 [-20–5]	85 [75-88]	**4** [2-6]	-7 [-9–3]	68 [54-74]
	Softmax	1 [0-1]	12 [9-14]	98 [97-99]	**1** [1-2]	**-26** [-28–21]	94 [92-96]	**3** [2-4]	**-15** [-19–6]	89 [85-92]	3 [2-5]	**-6** [-11–3]	90 [85-93]
	Logit	1 [0-2]	13 [9-14]	98 [95-99]	**1** [0-1]	**-26** [-28–22]	96 [90-97]	**3** [2-4]	**-15** [-19–6]	87 [84-92]	3 [2-5]	**-6** [-10–2]	89 [81-94]
	Energy	1 [0-2]	12 [9-14]	98 [95-99]	0 [0-1]	**-26** [-28–22]	96 [90-98]	2 [2-3]	**-15** [-19–6]	89 [85-92]	3 [2-5]	**-6** [-11–2]	88 [79-94]
	Energy-React	1 [0-2]	12 [9-14]	98 [95-99]	0 [0-1]	**-26** [-28–22]	96 [90-98]	2 [2-3]	**-15** [-19–6]	89 [85-92]	3 [2-5]	**-6** [-11–2]	88 [79-93]
	Mahalonobis	1 [0-3]	13 [9-15]	84 [76-95]	0 [0-0]	-27 [-28–22]	98 [94-99]	0 [0-1]	-17 [-22–8]	95 [87-96]	0 [-1-3]	-9 [-12–6]	76 [70-94]
	GMM	1 [0-3]	13 [9-15]	81 [49-86]	0 [0-0]	-27 [-28–23]	98 [98-99]	0 [0-2]	-17 [-22–7]	95 [77-96]	0 [-2-3]	-9 [-12–6]	74 [52-80]
	PCA	1 [0-3]	13 [8-15]	82 [69-90]	0 [0-0]	-27 [-28–22]	99 [98-99]	0 [0-1]	-17 [-22–8]	95 [85-97]	0 [-2-3]	-9 [-13–6]	77 [56-83]
		Art			Clipart			Product			Real World		
		OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑
OfficeHome	Deep-KNN	**13** [11-17]	**-15** [-21–5]	72 [65-77]	**14** [9-16]	**-28** [-31–15]	69 [62-78]	5 [3-7]	-12 [-13-2]	89 [85-94]	**4** [2-6]	-10 [-14-1]	92 [87-94]
	ViM	8 [6-12]	-21 [-27–4]	76 [69-80]	11 [6-15]	**-28** [-33–19]	68 [56-83]	4 [2-5]	-12 [-14-1]	90 [87-95]	**4** [2-4]	-11 [-13-1]	89 [87-94]
	Softmax	11 [8-16]	-17 [-23–8]	76 [66-86]	10 [8-16]	**-28** [-33–18]	75 [64-82]	**7** [3-8]	**-11** [-12-2]	87 [84-95]	**4** [3-8]	**-8** [-12-1]	93 [84-95]
	Logit	**13** [9-17]	-16 [-21–6]	73 [66-82]	11 [8-16]	**-28** [-35–15]	74 [63-81]	5 [2-8]	-12 [-13-2]	90 [83-95]	3 [2-8]	-9 [-14-1]	94 [83-96]
	Energy	12 [9-16]	-17 [-22–6]	75 [66-82]	10 [6-16]	-30 [-36–16]	74 [64-83]	4 [2-7]	-12 [-14-2]	91 [84-96]	3 [2-7]	-10 [-14-1]	95 [83-96]
	Energy-React	12 [9-16]	-17 [-22–6]	74 [65-82]	11 [7-16]	**-28** [-36–16]	74 [63-82]	5 [2-7]	-12 [-14-2]	91 [84-95]	3 [2-7]	-10 [-14-1]	95 [84-96]
	Mahalonobis	1 [0-12]	-28 [-33–4]	91 [70-93]	1 [0-6]	-40 [-43–21]	89 [75-92]	0 [0-2]	-16 [-18-1]	94 [93-95]	0 [0-2]	-15 [-17-1]	91 [87-95]
	GMM	1 [0-11]	-28 [-34–5]	92 [69-93]	0 [0-6]	-40 [-43–22]	90 [77-92]	0 [0-2]	-16 [-18-1]	95 [93-95]	0 [0-2]	-15 [-17-1]	91 [87-95]
	PCA	0 [0-8]	-28 [-34–8]	92 [77-95]	0 [-1-6]	-40 [-43–22]	92 [77-94]	0 [0-2]	-17 [-19-1]	95 [94-96]	0 [0-2]	-15 [-18-0]	92 [89-96]
		L100			L38			L43			L46		
		OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑
TerraIncognita	Deep-KNN	**34** [30-45]	**1** [-18-5]	31 [16-41]	**30** [18-38]	**-16** [-36–6]	45 [34-53]	**23** [11-34]	**-11** [-20–1]	44 [34-53]	42 [2-54]	-8 [-56-4]	14 [10-30]
	ViM	**34** [25-38]	-6 [-13-4]	32 [12-47]	23 [14-36]	-19 [-41–11]	51 [33-63]	17 [8-22]	-16 [-25–12]	47 [41-57]	**45** [20-55]	**-7** [-34-6]	13 [4-34]
	Softmax	4 [2-12]	-31 [-46–18]	84 [74-95]	6 [1-9]	-43 [-55–30]	83 [72-94]	3 [2-7]	-28 [-34–23]	91 [80-96]	4 [1-16]	-48 [-60–34]	78 [55-93]
	Logit	5 [2-20]	-30 [-45–12]	85 [56-96]	9 [2-16]	-39 [-54–23]	76 [65-93]	2 [1-6]	-30 [-33–25]	94 [82-98]	3 [1-19]	-47 [-61–31]	86 [50-94]
	Energy	5 [1-21]	-30 [-45–11]	86 [49-97]	10 [2-17]	-38 [-54–21]	75 [62-91]	2 [0-6]	-31 [-33–25]	94 [83-99]	3 [1-20]	-46 [-61–28]	84 [49-94]
	Energy-React	5 [2-21]	-30 [-45–12]	86 [50-97]	10 [2-17]	-38 [-54–21]	75 [62-91]	2 [0-6]	-31 [-33–25]	94 [83-99]	3 [1-21]	-46 [-61–28]	84 [49-94]
	Mahalonobis	6 [-2-24]	-26 [-48–14]	62 [23-91]	5 [1-20]	-38 [-55–28]	75 [57-94]	3 [0-12]	-28 [-36–19]	62 [37-90]	13 [1-44]	-35 [-56–7]	15 [3-92]
	GMM	10 [-3-27]	-25 [-45–9]	33 [19-77]	7 [1-28]	-37 [-49–21]	70 [45-89]	5 [0-12]	-28 [-34–19]	62 [37-72]	13 [3-44]	-34 [-54–7]	14 [3-54]
	PCA	0 [-2-11]	-36 [-48–20]	88 [50-99]	1 [0-18]	-44 [-53–28]	96 [57-99]	0 [-1-6]	-33 [-36–25]	90 [60-96]	4 [0-26]	-45 [-56–28]	72 [17-82]

Table C.1. Accuracy on competence region of OOD domain for different *PACS*, *OfficeHome*, *VLCS* and *TerraIncognita* domains and incompetence scores. As the threshold for the competence regions, we choose the 95% percentile of the ID validation set. For all metrics, a higher value means better performance (↑). All displayed values are medians over different domain roles and classifiers, brackets indicate 90% confidence interval.

C.6. Trained Classifiers

Figure C.13 shows the accuracies of all different Classifiers on all DG datasets for the ID data and the OOD data. Here we show the means and standard deviations over the different domains. All classifiers obtain a similar ID and OOD accuracy. One exception is VREx which did not converge for all domains on DomainNet. In Figure C.12 we show the accuracies for the different DG methods on all datasets.

In Percentages (%)	clip			info			paint		
	OOD-Gain ↑	ID-Gap ↑	Frac ↑	OOD-Gain ↑	ID-Gap ↑	Frac ↑	OOD-Gain ↑	ID-Gap ↑	Frac ↑
Deep-KNN	**4** [2-5]	**4** [2-7]	86 [82-90]	3 [0-6]	**-40** [-45–31]	81 [71-89]	**4** [1-6]	**-9** [-11–6]	86 [84-92]
ViM	**4** [2-6]	**4** [1-6]	90 [85-93]	1 [0-9]	-42 [-44–28]	92 [66-97]	2 [1-8]	-10 [-13–3]	92 [80-95]
Softmax	3 [1-3]	3 [1-5]	95 [94-97]	**5** [1-7]	**-40** [-42–28]	78 [74-88]	3 [1-4]	-10 [-12–8]	93 [91-95]
Logit	3 [1-4]	**4** [2-5]	92 [90-96]	2 [0-8]	-42 [-44–30]	87 [72-96]	3 [1-4]	-10 [-12–8]	93 [90-95]
Energy	3 [1-5]	3 [1-5]	91 [88-96]	1 [0-8]	-42 [-44–30]	92 [73-97]	3 [1-4]	-10 [-13–7]	93 [90-95]
Energy-React	3 [1-5]	3 [1-5]	92 [88-96]	1 [0-8]	-42 [-44–30]	92 [72-98]	2 [1-4]	-10 [-13–7]	93 [91-96]
Mahalonobis	-1 [-3-3]	-2 [-3-5]	86 [84-93]	0 [-1-4]	-44 [-45–32]	96 [79-97]	0 [-1-8]	-14 [-16–3]	95 [80-97]
GMM	-2 [-3-2]	-2 [-4-4]	87 [84-94]	0 [-1-3]	-45 [-46–32]	97 [83-98]	-1 [-1-6]	-14 [-16–4]	95 [80-97]
PCA	-2 [-3-1]	-2 [-4-3]	86 [84-93]	0 [-1-1]	-45 [-47–32]	96 [87-98]	-1 [-1-1]	-14 [-16–10]	96 [90-97]
	quick			real			sketch		
	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑
Deep-KNN	**3** [0-4]	**-48** [-52–27]	78 [65-91]	**3** [1-3]	**5** [2-9]	92 [89-95]	**6** [2-9]	**-4** [-6-0]	83 [79-87]
ViM	2 [0-3]	-49 [-54–27]	80 [65-99]	2 [0-4]	4 [2-11]	94 [88-97]	3 [0-6]	-7 [-9–1]	90 [86-95]
Softmax	0 [0-2]	-51 [-54–27]	94 [88-98]	1 [1-2]	3 [1-8]	97 [97-98]	3 [2-4]	-7 [-9–1]	93 [92-94]
Logit	1 [0-2]	-50 [-54–27]	89 [72-95]	2 [1-2]	3 [2-7]	97 [96-98]	3 [1-5]	-7 [-8–1]	93 [91-94]
Energy	1 [0-2]	-50 [-54–27]	88 [71-98]	1 [0-2]	3 [1-7]	98 [95-98]	3 [0-4]	-7 [-9–2]	94 [91-95]
Energy-React	1 [0-2]	-50 [-54–27]	88 [72-98]	1 [0-2]	3 [1-7]	98 [95-98]	3 [0-4]	-8 [-9–2]	94 [91-95]
Mahalonobis	0 [-1-1]	-52 [-55–27]	94 [68-100]	-1 [-3-5]	1 [-2-11]	87 [81-91]	-1 [-1-5]	-10 [-13–2]	93 [86-95]
GMM	0 [-1-0]	-52 [-55–27]	95 [75-100]	-1 [-3-4]	0 [-3-10]	87 [79-90]	-1 [-2-3]	-11 [-14–2]	93 [88-95]
PCA	0 [-1-0]	-52 [-55–27]	94 [75-99]	-2 [-3-3]	1 [-2-9]	87 [79-90]	-1 [-2-2]	-11 [-13–3]	93 [89-95]

Table C.2. Accuracy on competence region of OOD domain for different DomainNet domains and incompetence scores. As the threshold for the competence regions, we choose the 95% percentile of the ID validation set. For all metrics, a higher value means better performance (↑). All displayed values are medians over different domain roles and classifiers, brackets indicate 90% confidence interval.

In Percentages (%)	aclass			escape			hilux			i3			lexu		
	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Frac ↑
Deep-KNN	**2** [1-4]	**0** [0-0]	56 [19-64]	**14** [1-20]	**0** [-1-0]	41 [23-63]	**3** [1-8]	**0** [-1-0]	25 [19-56]	**17** [4-27]	**0** [-3-0]	15 [10-26]	**4** [1-9]	**0** [0-0]	27 [14-45]
ViM	**2** [1-4]	**0** [0-0]	42 [26-65]	11 [1-20]	**0** [-3-0]	28 [12-56]	**3** [1-7]	**0** [0-0]	17 [11-33]	**17** [4-28]	**0** [0-0]	12 [8-17]	**4** [2-9]	**0** [0-0]	20 [6-44]
Softmax	**2** [1-3]	**0** [-1-0]	82 [71-86]	11 [1-18]	**0** [-4-0]	67 [55-74]	2 [1-7]	**0** [0-0]	69 [57-72]	7 [-1-18]	-1 [-30-0]	51 [33-64]	3 [1-9]	**0** [-1-0]	56 [35-77]
Logit	1 [0-3]	**0** [-1-0]	81 [72-90]	7 [0-14]	**0** [-15-0]	73 [65-82]	2 [1-5]	**0** [-5-0]	73 [54-80]	4 [-2-18]	-2 [-30-0]	54 [35-79]	2 [1-8]	**0** [-3-0]	60 [36-81]
Energy	1 [0-3]	**0** [-1-0]	81 [72-90]	5 [-1-14]	**0** [-16-0]	73 [66-85]	2 [0-5]	**0** [-6-0]	73 [54-81]	4 [-2-18]	-2 [-30-0]	55 [35-79]	2 [1-7]	**0** [-3-0]	60 [36-81]
Energy-React	1 [0-3]	**0** [-1-0]	81 [72-90]	5 [-1-14]	**0** [-16-0]	73 [65-85]	2 [0-5]	**0** [-6-0]	73 [53-81]	4 [-2-17]	-2 [-30-0]	55 [35-79]	2 [1-8]	**0** [-3-0]	61 [34-81]
Mahalonobis	1 [0-3]	**0** [-1-0]	44 [23-94]	4 [-8-19]	-2 [-20-0]	19 [17-94]	2 [-1-7]	0 [-5-0]	16 [8-95]	4 [-17-28]	**0** [-32-0]	18 [6-93]	3 [0-9]	**0** [-3-0]	20 [7-95]
GMM	1 [1-3]	**0** [-1-0]	44 [24-70]	3 [-8-19]	-1 [-20-0]	19 [17-61]	2 [-1-7]	**0** [-4-0]	16 [8-42]	7 [-16-28]	**0** [-30-0]	15 [6-19]	2 [1-9]	**0** [-4-0]	21 [7-47]
PCA	0 [0-1]	-1 [-3-0]	89 [68-99]	-1 [-2-7]	-15 [-22-0]	83 [67-93]	1 [0-6]	-1 [-4-0]	84 [42-94]	0 [-2-3]	-19 [-28-0]	90 [48-100]	0 [0-2]	-3 [-9–1]	88 [66-95]
	tesla			tiguan			tucson			x5			zoe		
	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑	OOD-Gain ↑	ID-Gap ↑	Coverage ↑
Deep-KNN	**21** [2-36]	**0** [-1-0]	5 [2-18]	**1** [1-4]	**0** [0-0]	43 [22-66]	**2** [0-4]	0 [0-0]	37 [19-66]	**20** [13-32]	**0** [0-0]	16 [5-41]	**13** [2-25]	**0** [-1-0]	28 [18-46]
ViM	**21** [2-37]	**0** [0-0]	7 [2-17]	**1** [1-4]	**0** [0-0]	38 [19-67]	**2** [0-4]	0 [0-0]	**20** [14-65]	20 [13-31]	**0** [0-0]	11 [5-23]	**13** [2-25]	**0** [0-0]	19 [9-43]
Softmax	17 [-2-27]	-1 [-28-0]	35 [22-45]	**1** [0-3]	**0** [0-0]	68 [52-84]	**2** [0-4]	0 [0-0]	63 [39-81]	**20** [13-30]	-1 [-1-0]	42 [26-56]	9 [0-25]	**0** [-12-0]	55 [36-70]
Logit	4 [-9-21]	-2 [-46-0]	60 [26-72]	**1** [1-2]	**0** [-2-0]	79 [56-90]	**2** [0-4]	0 [-1-0]	73 [52-88]	19 [9-28]	-2 [-8-0]	54 [41-66]	9 [-1-21]	**0** [-16-0]	57 [43-71]
Energy	4 [-9-20]	-2 [-46-0]	60 [26-74]	**1** [0-2]	**0** [-2-0]	79 [57-90]	**2** [0-4]	0 [-2-0]	74 [59-88]	19 [1-28]	-2 [-17-0]	56 [42-66]	5 [-1-21]	**0** [-22-0]	57 [50-71]
Energy-React	4 [-11-20]	-2 [-48-0]	60 [27-74]	**1** [0-2]	**0** [-2-0]	79 [66-91]	**2** [0-4]	0 [-2-0]	74 [59-88]	19 [1-27]	-3 [-17-0]	56 [45-66]	5 [-2-21]	**0** [-23-0]	57 [48-72]
Mahalonobis	20 [0-37]	-1 [-4-0]	7 [2-94]	**1** [0-3]	**0** [-1-0]	35 [19-95]	**2** [0-4]	0 [-2-0]	23 [9-95]	18 [-14-31]	-6 [-33-0]	12 [6-95]	6 [-4-21]	**0** [-24-0]	25 [10-93]
GMM	19 [2-37]	**0** [-4-0]	7 [3-22]	**1** [1-3]	**0** [0-0]	35 [19-71]	**2** [0-4]	0 [0-0]	22 [9-69]	18 [-15-31]	**0** [-32-0]	13 [6-24]	9 [-6-25]	**0** [-14-0]	26 [10-39]
PCA	2 [0-32]	-6 [-25–1]	83 [32-91]	0 [0-2]	-1 [-3-0]	90 [70-97]	1 [0-4]	0 [-3-0]	90 [66-96]	7 [-2-17]	-12 [-30–1]	80 [48-94]	0 [0-19]	-5 [-21-0]	91 [57-100]

Table C.3. Accuracy on competence region of OOD domain for different SVIRO domains and incompetence scores. As the threshold for the competence regions, we choose the 95% percentile of the ID validation set. For all metrics, a higher value means better performance (↑). All displayed values are medians over different domain roles and classifiers, brackets indicate 90% confidence interval.

ID dataset	Open world dataset	Test domain → Open world domains	Open world classes
PACS	DomainNet	art → painting cartoon → clipart photo → photo sketch → sketch	alarm clock, ambulance, apple, backpack, baseball, basketball, bat, bear,bed and bicyle
VLCS	PACS	all environments → photo	elephant, giraffe and guitar
OfficeHome	DomainNet	art → paint clipart → clipart product → real real world → real	bread, butterfly, cake, carrot, cat
TerraIncognita	PACS	all enviroments → photo	elephant, giraffe and horse

Table C.4. Open world extensions of different DG datasets and their test domains.

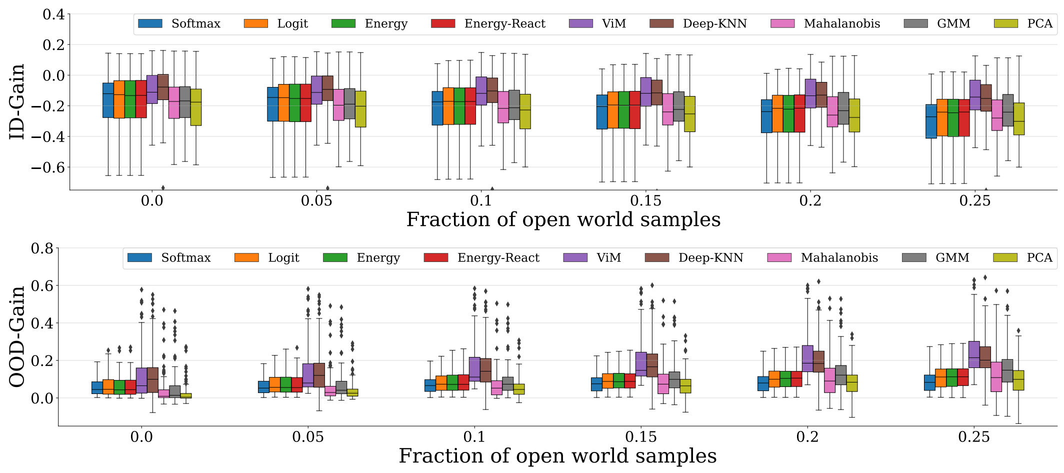


Figure C.10. *OOD-Gain* and *ID-Gain* for different incompetence score for an increasing fraction of open world data (unknown classes) in the test domain (higher is better).

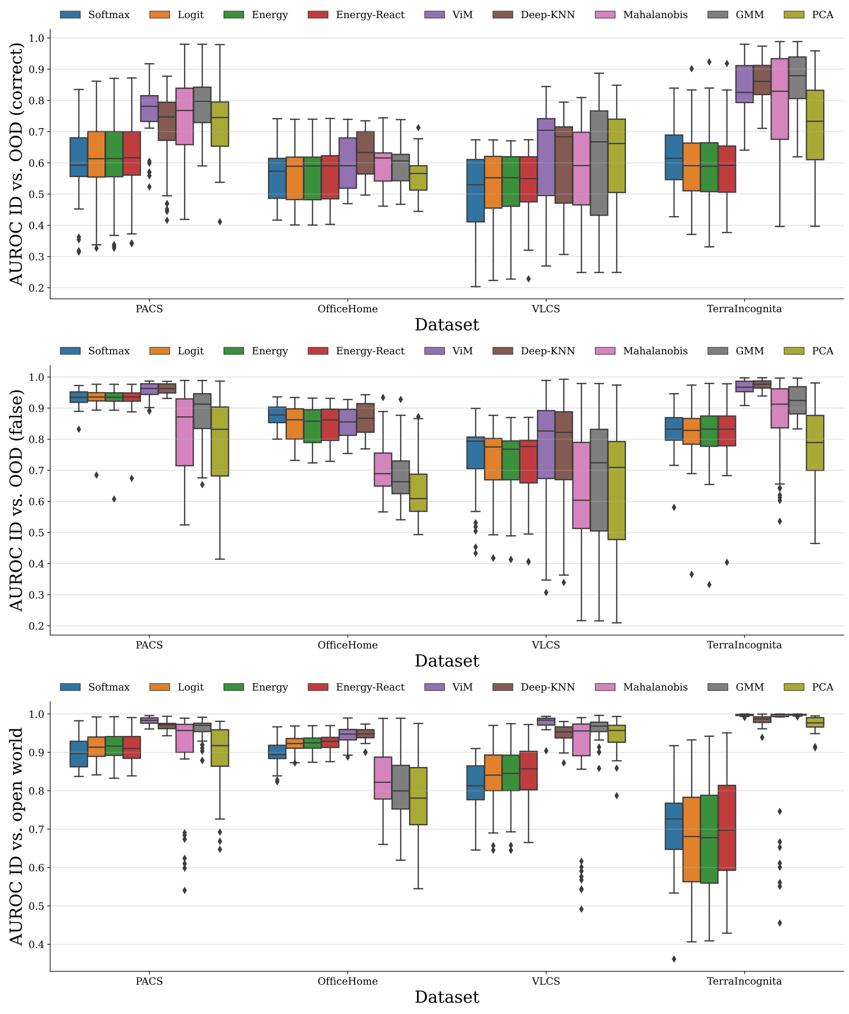


Figure C.11. *Above:* AUROC of delineating ID data vs. correctly classified samples on the OOD data. *Middle:* AUROC of delineating ID data vs. wrongly classified samples on the OOD data. *Below:* AUROC of delineating ID data vs. open world data in general. All test domains are enriched with 25% open world outliers.

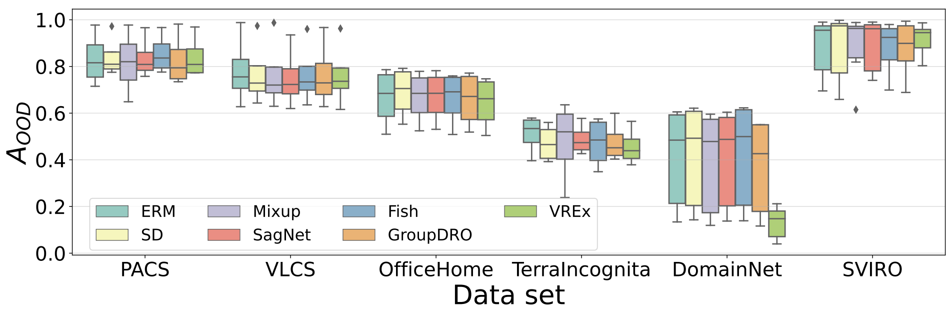


Figure C.12. Accuracies for different classifiers on OOD test data. The boxes show the quartiles and medians.

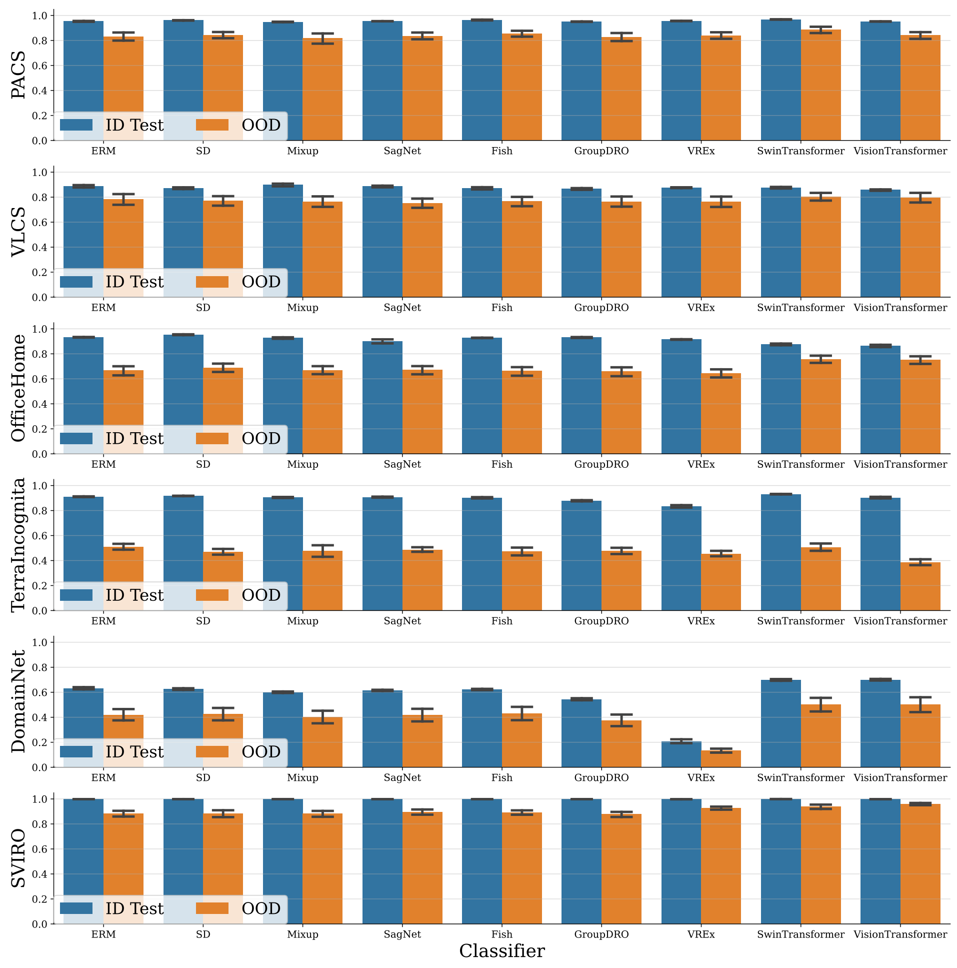
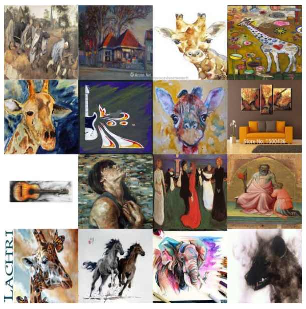
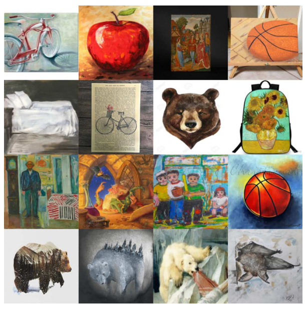


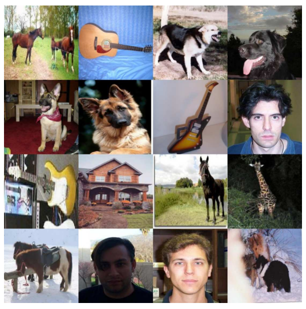
Figure C.13. Accuracies for different classifiers on ID and OOD test data. We show the means and standard deviations over different DG tasks.


(a) Test data for domain Art


(b) Open world data for domain Art

(c) Test data for domain Cartoon

(d) Open world data for domain Cartoon.


(e) Test data from domain Photo.

(f) Open world data for domain Photo.

(g) Test data from domain Sketch.

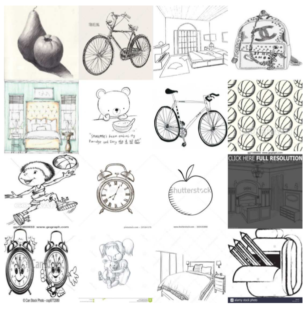
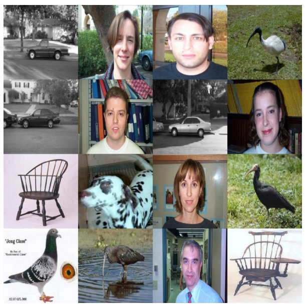
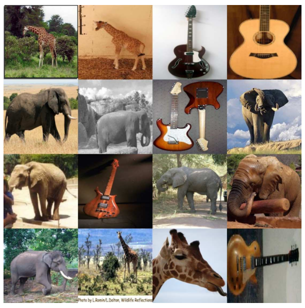

(h) Open world data for domain Sketch.

Figure C.14. Test (left) and open world data (right) for the PACS dataset.

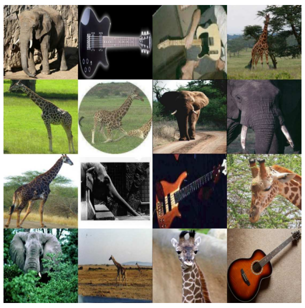


(a) Test data from domain Cal101.



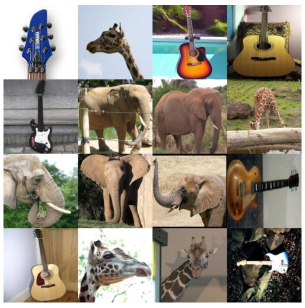
(b) Open world data for domain Cal101.

(c) Test data from domain LabelMe.

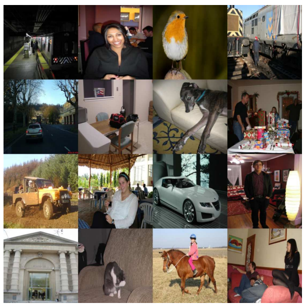


(d) Open world data for domain LabelMe.

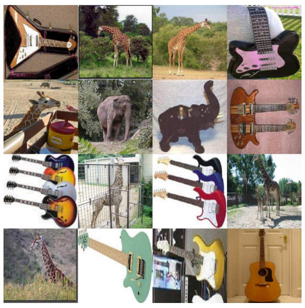
(e) Test data from domain SUN09.



(f) Open world data for domain SUN09.

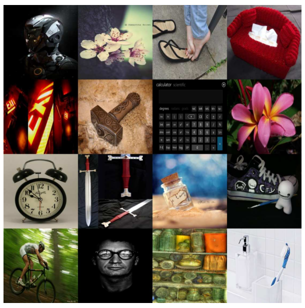


(g) Test data from domain VOC2007.



(h) Open world data for dom. VOC2007.

Figure C.15. Test (left) and open world data (right) for the VLCS dataset.


(a) Test data from domain Art.

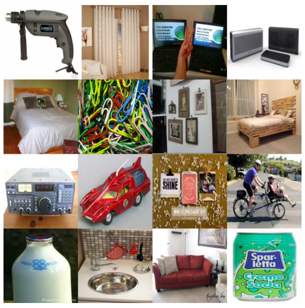
(b) Open world data for domain Art.

(c) Test data from domain Clipart.

(d) Open world data for domain Clipart.

(e) Test data from domain Product.

(f) Open world data for domain Product.


(g) Test data from domain Real World.

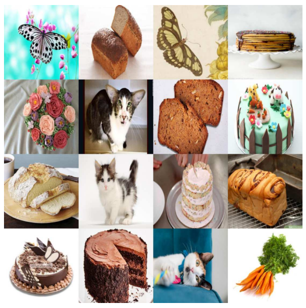
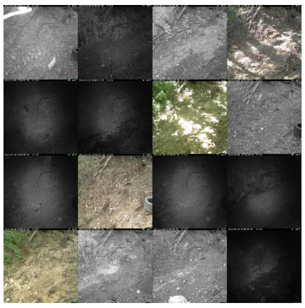
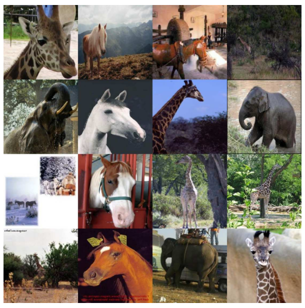

(h) Open world data for dom. Real World.

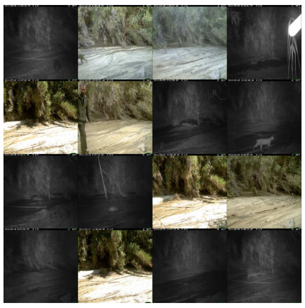
Figure C.16. Test (left) and open world data (right) for the OfficeHome dataset.

(a) Test data from domain L100.

(b) Open world data for domain L100.


(c) Test data from domain L38.


(d) Open world data for domain L38.


(e) Test data from domain L43.

(f) Open world data for domain L43.

(g) Test data from domain L46.

(h) Open world data for domain L46.

Figure C.17. Test (left) and open world data (right) for the TerraIncognita dataset.