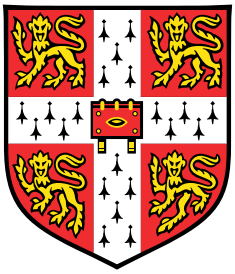
Learning Transferable Representations



Mateo Rojas-Carulla

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Corpus Christi College September 2018

A papá, mamá y Miguel, mi más grande tesoro.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, bibliography, footnotes, tables and equations and has fewer than 150 figures.

Mateo Rojas-Carulla
September 2018

Acknowledgements

The journey towards this thesis would not have been possible without the outstanding scientists and friends that I encountered along the way.

Bernhard Schölkopf provided me with encouragement through stressful periods, gave me constant support and freedom to pursue open research questions. *Richard Turner* taught me how to tackle a scientific question in a principled manner; his optimism and kindness are unmatched and always made things better. *Jonas Peters* was key in shaping my critical thinking and made a better scientist out of me, I am forever grateful.

Ilya Tolstikhin, a dear friend and mentor, helped make my stay at the Max Planck into some of the best years of my life. His desire to learn and improve, as well as his dedication to family and friends, make him a role model in science and in life.

I have been fortunate to be surrounded by amazing people who have contributed in ways which are hard to express. *Niki Kilbertus* and *Giambattista Parascandolo* taught me that science benefits from a great team and that fun and curiosity should drive research. *Sebastian Gómez*'s optimism and values were always an inspiration and provided much needed support.

I am thankful to *David Lopez-Paz*, from whom I learned scientific practices I now follow, and *Marco Baroni*, who was kind enough to provide his time and invaluable feedback.

Cambridge and the Max Planck are wonderful first and foremost because of their outstanding members, who have contributed to make me into a better person and a better scientist, including *Diego Agudelo, Carl-Johann Simon-Gabriel, Niklas Pfister, Paul Rubenstein, Stefan Bauer, Dieter Büchler, Okan Koc, Yassine Nemmour, Vinay Jayaram, Eduardo Pérez, the CamTue crowd*, and many others.

Receiving my PhD would not have been possible without people who taught me the foundations and ignited my desire to learn and keep asking questions. I would particularly like to thank *Stephan Céroi*, who gave me a strong mathematical foundation in high school, *Régine Astruc*, a great teacher who supported me during the difficult yet exciting years of prépa, and *Jean-Luc Vidal*, who provided invaluable guidance.

I am forever grateful to Kata, who put up with me during the writing of this thesis and has brought so much wonder and joy into my life. Last but not least, I dedicate this thesis to my amazing family, to whom I owe everything I have achieved and who bring to life the meaning of unconditional love.

Abstract

A first contribution of this thesis is to propose causality as a language for problems of distribution shift. First, we consider domain generalisation, where no data from the test distribution are observed during training. What assumptions can be made regarding the relation between train and test distributions for transfer to succeed? We argue that assuming the data in both tasks originate from the same causal graph leads to a natural solution: use only causal features for prediction, as the mechanism mapping causes to effects is invariant to shifts in the probability distributions induced by the causal structure. We provide optimality results when the test task is adversarial, and introduce a method for exploiting all remaining features when data from the test task are observed. We motivate that learning such invariant mechanisms mapping features to outputs leads to machine learning modules robust to transfer.

Second, we consider a classification problem where only few examples are available for each label. How should an initial large dataset be leveraged to improve performance in this task? We argue that such a dataset should be used to learn powerful features for batch classification using a neural network. We present a framework which transfers between classes by building a probabilistic model on the weights of the network. Our results suggest that practitioners should use the original dataset for building features whose power can be exploited during few-shot learning.

Finally, we extend causal discovery to solve problems such as distinguishing a painting from its counterfeit. Given two such static entities, a proxy random variable introduces the randomness necessary to construct two features of the static entities which preserve their causal footprint, measurable by a standard causal discovery procedure. Experiments on vision and language provide evidence that the causal relation between the static entities can often be identified.

Table of contents

List of figures

List of tables

List of notations

Abbreviations

DAG	Directed Acyclic Graph.
SEM	Structural equation model.
OCD	Observational Causal Discovery.
ANM	Additive noise model.
DS	Dataset shift.
CS	Covariate shift.
MMD	Maximum Mean Discrepancy.
DG	Domain Generalisation.
MTL	Multi-task Learning.
CNN	Convolutional Neural Network.
MAP	Maximum A Posteriori.
RKHS	Reproducing Kernel Hilbert Space.
GAN	Generative Adversarial Network.
GMM	Gaussian Mixture Model.

General notations

$\mathbb{R}^d$	d dimensional real vector space.
$\mathbb{R}^+$	Non-negative real numbers.
$\mathcal{X}$	Set.
$\langle .,. \rangle$	Dot product.

Probability

X	Real valued random variable.
$\mathbf{X}$	Vector valued random variable.
$P(X)$	Probability distribution of random variable X.
$p := p(X)$	Probability density function of random variable X.
$\mu_\kappa(P)$	Mean embedding of distribution P in the RKHS defined by kernel κ.
$\overset{\mathrm{d}}{=}$	Equality in distribution.

Statistics

i.i.d.	Independent and identically distributed.
$\{x_i\}_{i=1}^n \overset{iid}{\sim} P(X)$	Sample drawn i.i.d. from $P(X)$.

Chapter 1

Introduction

Machine learning provides tools for making predictions in the world around us. When learning with supervision, a machine or agent is given a training dataset of input-output pairs $(\mathbf{x}, y)$, where $\mathbf{x} \in \mathcal{X}$ is a *feature representation* of the data point, and y the output value corresponding to the input $\mathbf{x}$. The goal of the agent is to build a *model* M allowing us to make *predictions* on new test data points $\widetilde{\mathbf{x}}$. It remains nonetheless unclear where the new data point $\widetilde{\mathbf{x}}$ is coming from, and whether M can be used for prediction without further modifications. Most standard approaches for supervised learning assume that the test data is drawn from the same distribution P as the training data, thus circumventing this problem: when testing on data coming from P, M may be deployed directly.

In practice, however, this assumption is often violated for a variety of reasons. The test data may have been acquired on a subset of the population which is not well represented in the training set. The labels present in the test set may be different from those abundant in the training set. A robot having learned to play tennis well should in principle be able to transfer knowledge to sports sharing some of the same skills, such as squash. In all these situations and many more, we wish to learn models which can be *transferred* to the test distribution.

A fundamental problem when learning a predictive model M is to learn a feature representation

$$h : \mathcal{X} \to \widetilde{\mathcal{X}}$$

of the input data which better expresses the key attributes of the input necessary to solve the task at hand. In standard multi-class image classification, the input to the model is usually a vector of pixel intensities $\mathbf{x} \in \mathbb{R}^d$ from which distinguishing labels is a difficult task. From the machine's perspective, $\mathbf{x}$ is a vector of numbers with little apparent regularity. Simply rotating the image leaves the output trivially recognisable for humans, yet the resulting pixel intensities look very different from the original $\mathbf{x}$. Powerful machine learning models such as kernel methods (Schölkopf et al., 2002) and deep neural networks (Goodfellow et al., 2016)

build a linear model on top of a learned feature representations $h(\mathbf{x})$, providing evidence that it is possible to learn informative features leading to high classification accuracy with linear models. One can understand, for instance, that a binary vector where each dimension indicates if an attribute is present in the image (Is there a tail? Is there a chair?) is significantly simpler to exploit for classification than the raw pixel representation. This thesis analyses different problems in machine learning in which transfer occurs, and introduces methods for learning appropriate representations and models.

But what kind of assumptions regarding the similarity between different distributions can be made? A main contribution of this thesis is the formulation of regularity assumptions between training and test distributions using ideas from the field of causal inference, see Chapter 3. We motivate that learning representations which exploit causal properties of the data generating process leads to attractive properties for transfer when the training and test distributions shift. One fundamental axis of causality is thinking about how the data came to be, what are the *mechanisms* underlying the generation of the data, which in turn explain the statistical dependencies we measure. Exploiting such dependencies provides a powerful tool for prediction under the same training and testing conditions, but can easily break under distribution shift. We argue that only *causal* features of the target variable should be used for prediction when no data from the test task are observed. This is an application of the principle of independence between cause and mechanism (Peters et al., 2017): the function or mechanism transforming causes to effect remains invariant to a change in the distribution of the causes. Such an invariant mechanism mapping causes to effect can thus be re-used for prediction in distributions which are drawn from the same causal graph, even when the statistical properties of such distributions differ. A first objective of this thesis is thus to show that *invariant causal features provide a powerful framework to discuss distribution shift problems.* We further exploit this principle when a probability distribution P is modified by several independent functions. This is the case, for instance, when a distribution over natural images is transformed by different noisy processes and geometric transformations. We show that given data from these transformed distributions, we can recover an approximate mapping back to the original distribution P without supervision, and the learned mechanisms generalise to new distributions altogether. This result further supports the re-usability of models resulting from causal assumptions.

The success of deep learning methods in supervised learning problems requires large amounts of labelled data. Often, as providing labels for a dataset is expensive, only a handful of examples are available and these methods cannot be applied. The problem of few-shot learning receives as input an initial large dataset abundant in labelled data, and a small dataset containing new classes that are not present in the training set. From the perspective of human learning, recognising new entities in the world around us seems to leverage two key properties: first, we extract powerful visual features allowing us to focus on high level attributes of the entities we want to recognise; and second, we are able to transfer information

from similar entities we have encountered in the past. Learning to recognise a new type of flower is made easier when we have seen many types of flowers before and are aware of the features they share, and the ones which differentiate them. Inspired by this observation, we propose to build representations using an initial large dataset with standard deep learning approaches. If the new classes are similar to the training classes, we show empirically that these representations transfer well and allow us to quickly learn new concepts, achieving state-of-the-art results in the *mini*ImageNet dataset. A second goal of this thesis is thus to *provide a framework for few-shot learning and motivate that better features for batch classification transfer well to new few-shot learning classes.*

Causal discovery algorithms aim to find the causal direction between variables given a sample from their joint distribution. Inferring causality from purely observational data is often not possible, and assumptions about the data generating process must be made for a *causal footprint* to become identifiable. We present an extension of causal discovery motivated by the following question: how can you teach a machine that the concept "virus" *causes* the concept "death"? The causal direction is quite clear to us, but is based on an understanding of biological processes, and how untreated infection can lead to death. From the point of view of causal discovery, obtaining a sample from two such *static* entities x and y which could be fed to a causal discovery algorithm is an ill-defined problem. Using an additional source of randomness, we propose to build representations of x and y from which we can infer the causal direction. A third objective of this thesis is thus to show empirically that *we can construct representations which transfer the causal footprint existing between the original static entities.* We introduce a dataset of human-designed, human-validated pairs of concepts, and show that we can correctly identify over 75% of the causal directions using a causal discovery algorithm.

1.1 Outline and Contributions

We provide an outline of this thesis, as well as the major contributions.

- Chapter 2 provides a brief introduction to causal inference and discovery, and describes key properties which are exploited later in the thesis. In particular, Section 2.3 introduces the idea of invariant causal conditionals, and summarises some results on the identifiability of the causes when data from different tasks are available. These ideas motivate the methods for domain shift introduced in Chapter 3. Section 2.2 introduces observational causal discovery, and methods for estimating the causal direction between two variables given only an observational sample from their joint distribution. These methods are used to identify the causal direction between static entities in Chapter 5.

- Chapter 3 focuses on the distribution shift problem. We introduce a relaxation of covariate shift motivated by the notion of invariant causal features. Section 3.2 discusses

the setting in which no data from the test task are observed. We motivate that using only *invariant* features is optimal when the new task is chosen in an adversarial way. Section 3.2.6 introduces a method for estimating an invariant subset from data. In Section 3.3, we also provide a method to exploit this knowledge for prediction when some data from the test task are observed. Both these scenarios are discussed in Rojas-Carulla et al. (2018). Section 3.4 discusses a specific application of multi-task learning in which we want to identify and invert a set of functions automatically and follows Parascandolo et al. (2018). We show that MNIST digits transformed by a series of geometric and noisy transformations can be mapped back to their canonical counterpart without supervision. We further show that the learned mappings generalise to unseen datasets.

- Chapter 4 expands on Bauer et al. (2017). We propose a framework for few-shot learning based on two insights: using features learned from a large image dataset makes learning at few-shot time significantly easier, and the weights of the network contain information about the classes and can be used to regularise the weight vectors of the new classes. Our probabilistic method achieves state-of-the-art in a standard dataset for few-shot learning, putting into context recent developments in the field arguing for meta-learning methods.

- Chapter 5 focuses on the problem of causal discovery between two *static* entities, and is based on Rojas-Carulla et al. (2017). How do we apply a causal discovery algorithm to discover the relation between a painting and its counterfeit? In such case, only one example of each paining is available, while a key ingredient of causal discovery methods is having access to a large sample from which a causal footprint can be measured. We introduce the notion of proxy variable to build representations of the static entities from which the causal direction can be inferred. To assess the methods, we construct a large dataset of pairs of causal concepts, such as “virus” and “death”.

All the work was done in collaboration with several colleagues, including Jonas Peters, Richard Turner and Bernhard Schölkopf. Chapter 4 is based on equal contribution work with Matthias Bauer, but I did not work on designing and training the neural networks for batch classification. Section 3.4 is based on work to which I contributed in ideas and part of the writing, but not in the implementation.

The papers cited previously are mentioned below.

Rojas-Carulla, M., Schölkopf, B., Turner, R., and Peters, J. (2018). Invariant models for causal transfer learning. *Journal of Machine Learning Research*

Bauer, M., Rojas-Carulla, M., Świątkowski, J., Schölkopf, B., and Turner, R. (2017). Discriminative k-shot learning using probabilistic models. *Bayesian Deep Learning Workshop of the 31st Conference on Neural Information Processing Systems (NIPS)* (equal contribution)

Parascandolo, G., Kilbertus, N., Rojas-Carulla, M., and Schölkopf, B. (2018). Learning independent causal mechanisms. *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 80:4033 – 4041

Rojas-Carulla, M., Baroni, M., and Lopez-Paz, D. (2017). Causal discovery using proxy variables. *Workshop of the International Conference on Learning Representations (ICLR)*

During my PhD, I also worked on the following paper but do not discuss it in the thesis.

Kilbertus, N., Rojas Carulla, M., Parascandolo, G., Hardt, M., Janzing, D., and Schölkopf, B. (2017). Avoiding discrimination through causal reasoning. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, pages 656 – 666

Chapter 2

An Overview of Causal Inference

"I would rather discover a single causal law than be king of Persia". These words by the Greek philosopher Democritus attest to the long and challenging pursuit led by scientists and philosophers in the field of causality. From a scientific perspective, being able to establish with confidence the causal laws relating variables of interest remains a fundamentally difficult question. It is nonetheless true that the scientific community is more often than not interested in questions of a causal nature. What are the main causes of the obesity epidemic? What is the effect of increasing interest rates on economic growth after a recession? These questions are difficult because they require significantly more understanding of the true underlying mechanisms than traditional predictive modelling. Indeed, performing an action, or *intervention*, on the system of interest usually changes the distribution of the data, and insights resulting from statistical dependencies previously measured should be carefully revisited. It may be, for instance, that the amount of saturated fat in a person's diet is highly predictive of obesity. Concluding that saturated fat must be restricted solely on this premise is a significant leap of faith with remarkable public health consequences. While the hypothesis that high saturated fat causes obesity is valid, it is only one of many possible explanations. It may be, for instance, that the amount of saturated fat is simply confounded by total calories consumed, which fully explains obesity. It may also be that sugar is the culprit, and replacing sugar calories for saturated fat calories at constant total calories decreases a person's weight. In order to test these assumptions, careful large scale experiments must be carried out.

The gold standard for establishing causal links are randomised experiments. Consider the important problem of determining the effect of a new drug on recovery. The ideal experiment would take the same sick patient, clone her, and apply the treatment only to one of the copies. If only the treated individual recovers, the drug is a success. Since this imaginary intervention is not possible, randomised experiments replace this idealised experiment by a large scale experiment on subjects representative of the population in which the drug is to be prescribed. This population is randomly divided into two groups, a *control* group and an *experimental* group. The first group is given a placebo (sugar pill) and the second group

the drug under consideration. The randomised aspect of the experiment is essential, as it removes any possible effect of the treatment assignment on recovery (for example, the doctor only assigning the drug to rich people, who recover more often anyway). If the difference in recovery rates between both groups is statistically significant, the effect of the drug is confirmed. It is important to realize that without this experiment, many possible explanations for recovery cannot be ruled out with confidence.

While randomised experiments are a powerful tool for establishing causal links, they are often infeasible for several reasons. A first reason is ethics. A researcher aiming to establish the effect of abortions on some indicator of the woman's future well-being will certainly never carry out a randomised experiment. A second reason may be the pure impossibility of carrying out an experiment: finding a population of countries willing to have their interest rates tempered with is out of the question. The area of causal discovery emerged from statistics and econometrics to tackle this issue. What can we say about the causal link between quantities of interest based on the available data and the statistical dependencies we measure, without carrying out experiments? Section 2.2 introduces observational causal discovery, which aims to answer this question when the data available is purely observational and i.i.d. The methods discussed are the causal discovery black-boxes used in Chapter 5. Section 2.3 further assumes that some experimental data is available, and introduces the ideas which constitute the backbone of the methods for transfer learning presented in Chapter 3.

2.1 Fundamental Technical Notions

2.1.1 From Statistical to Causal Dependence

Let $\mathbf{X} = (X_1, \ldots, X_p)$ be p random variables. Predictive modelling relies on statistical dependencies, that is, properties of the joint distribution $P(X_1, \ldots, X_p)$ are exploited for prediction. Given a features $\mathbf{X}$ and a target variable Y we wish to predict from $\mathbf{X}$, regression models capture this dependency and build a model $\hat{Y} = f(\mathbf{X})$. However, a statistical dependence does not in general allow us to establish the causal relation between the variables in $X_1, \ldots, X_p$ and Y. Reichenbach (1956) attempted to formalise the relation between dependence and causation:

Principle 1 (Principle of Common Cause) *If two random variables X and Y are statistically dependent ($X \not\perp\!\!\!\perp Y$), then one of the following causal explanations must hold:*

i) X causes Y (write $X \to Y$), or

ii) Y causes X (write $X \leftarrow Y$), or

iii) there exists a random variable Z, such that $X \leftarrow Z \to Y$. Such a variable Z is called a confounder.

In the third case, X and Y are conditionally independent given Z (write $X \perp\!\!\!\perp Y \mid Z$).

In practice, dependencies can be observed between random variables for different reasons that the ones mentioned in Principle 1, and some examples are given in (Peters et al., 2017, Chapter 1.4). Nonetheless, the implications of Principle 1 are strong when we wish to perform interventions in a system and not only predict, and are in line with the well known warning *correlation does not imply causation*. As a classic example, consider Messerli (2012), where the authors find a significant correlation between $X =$ `Chocolate consumption`, measured per capita, and $Y =$ `Number of Nobel prize winners`. A common reaction to this finding often reads beyond the measured statistical dependency and attributes a causal meaning. In other words, people assume that *intervening* by eating more chocolate will lead to increased intelligence and thus, a higher probability of one day obtaining a Nobel prize. Even the authors of the paper write "since chocolate consumption has been documented to improve cognitive function, it seems most likely that in a dose-dependent way, chocolate intake provides the abundant fertile ground needed for the sprouting of Nobel laureates. Obviously, these findings are hypothesis-generating only and will have to be tested in a prospective, randomised trial." This hypothesis, however, seems far fetched and quite unlikely. Principle 1 provides a more likely explanation: there is a confounding variable Z which causes both X and Y and explains the measured dependence. Z could be, for instance, a country's wealth which partly explains a larger disposable income and better quality education. In the absence of a causal link from X to Y, intervening on the amount of chocolate eaten will have no effect on Y. Similar statements are commonplace in the media and highlight the difference between *observing* and *doing* (Pearl, 2009). We present some tools from the field of causal inference developed to discuss interventions and the resulting distribution shifts.

2.1.2 Directed Acyclic Graphs, Structural Equation Models

The causal relation between the variables $X_1, \ldots, X_p$ is often represented by a causal Directed Acyclic Graph (DAG) $\mathcal{G}$. An arrow $X_i \to X_j$ is present in $\mathcal{G}$ if X_i is a *direct cause* of X_j. A DAG entails a *causal factorisation* of the joint distribution $P(\mathbf{X})$:

$$P(X_1, \ldots, X_p) = \prod_{j=1}^{p} P(X_j \mid \mathbf{PA}_j^{\mathcal{G}}), \tag{2.1}$$

where $\mathbf{PA}_j^{\mathcal{G}}$ is the set of parents of X_j in $\mathcal{G}$. Let us consider a simple example with two random variables X and Y where X is a cause of Y. Standard statistical analysis does not a priori prefer any factorisation of the join distribution, since $P(X, Y) = P(Y \mid X)P(X) = P(X \mid Y)P(Y)$ are both valid equivalent factorisations. Only the first factorisation, however, is a causal factorisation as in (2.1), and its modules can easily be interpreted when discussing interventions on X and Y. As a concrete example, assume that $X =$ `time from birth`

and $Y = \texttt{height}$, where X is clearly a cause of Y. Given a young subject, a hypothetical intervention which fast-forwards time and therefore ages her 10 years should also imply an increase in height, just as transporting her back to her baby years will make her smaller. If one could, on the other hand, intervene on height, there is absolutely no reason to expect a movement in time. Despite both interventions being impossible in practice and fantastic in nature, we can reason about them because of our knowledge of the biological mechanism relating age to height.

Independence of cause and mechanism In the factorisation $P(X, Y) = P(Y \mid X)P(X)$ in the previous example, the distribution of the cause X is *independent* of the conditional $P(Y \mid X)$. The opposite, however, does not hold: a change in the distribution $P(Y)$ often modifies $P(X \mid Y)$. This is an instance of the Principle of Independent Mechanisms, which Peters et al. (2017) define as follows:

Principle 2 (Independent Mechanisms) *The causal generative process of a system's variables is composed of autonomous modules which do not influence each other. In the probabilistic case, this means that the conditional distribution of each variable given its causes does not inform or influence the other conditional distributions.*

As mentioned in Peters et al. (2017), this principle is plausible when the conditionals in (2.1) correspond to physical mechanisms in the world. Some attempts to formalize further the notion of independent mechanisms has been proposed in Janzing and Schölkopf (2010) using the notion of Kolmogorov complexity. However, the problem of quantifying independence between mechanisms given data remains an open research question.

Structural equation models The causal relation between random variables can be described by a Structural Equation Model (SEM). Although different definitions exist, we use the one from Peters et al. (2014).

Definition 3 *A Structural Equation Model is a tuple* $(\mathcal{E}, P(\mathbf{N}))$*, where* $\mathcal{E} = (E^1, \ldots, E^p)$ *is a set of* p *equations:*

$$E^j : \quad X^j = f^j(\mathbf{PA}_j^{\mathcal{G}}, N^j), \quad j = 1, \ldots, p. \tag{2.2}$$

$P_{\mathbf{N}}$ *is the joint distribution of the noise variables* $\mathbf{N}$*, which are assumed to be jointly independent. The vertices of* $\mathbf{PA}^j$ *are the parents of* X^j *in the corresponding DAG* $\mathcal{G}$*.*

Each equation E_j for $j \in \{1, \ldots, p\}$ specifies the functional relation between X_j and its parents. Standard methods for learning a causal graph from observational data include conditional independence tests (Zhang et al., 2011) and additive noise models (Peters et al., 2014).

Importantly, the equality in (2.2) should be interpreted as an assignment rather than an algebraic equality: E_j allows us to compute X_j from changes in its parents, but modifying

X_j does not trickle up to its parents in $\mathcal{G}$. In other words, modifying an equation in the SEM has an effect on the distribution of its descendants, but not its parents. This leads us to the definition of interventions.

Definition 4 (Interventions) *A hard intervention on X_j consists on setting $E_j : X_j = x$ for some x and using the SEM to recompute the value of all the descendants of X_j. Similarly, a soft intervention sets $E_j : X_j = \widetilde{N}$ for some random variable $\widetilde{N}$.*

A hard intervention $X_j = x$ is refered in Pearl (2009) as a *do* operation and is denoted $\text{do}(X_j = x)$. Similar notations are proposed for soft interventions.

Simple example. Consider the following simple bivariate example:

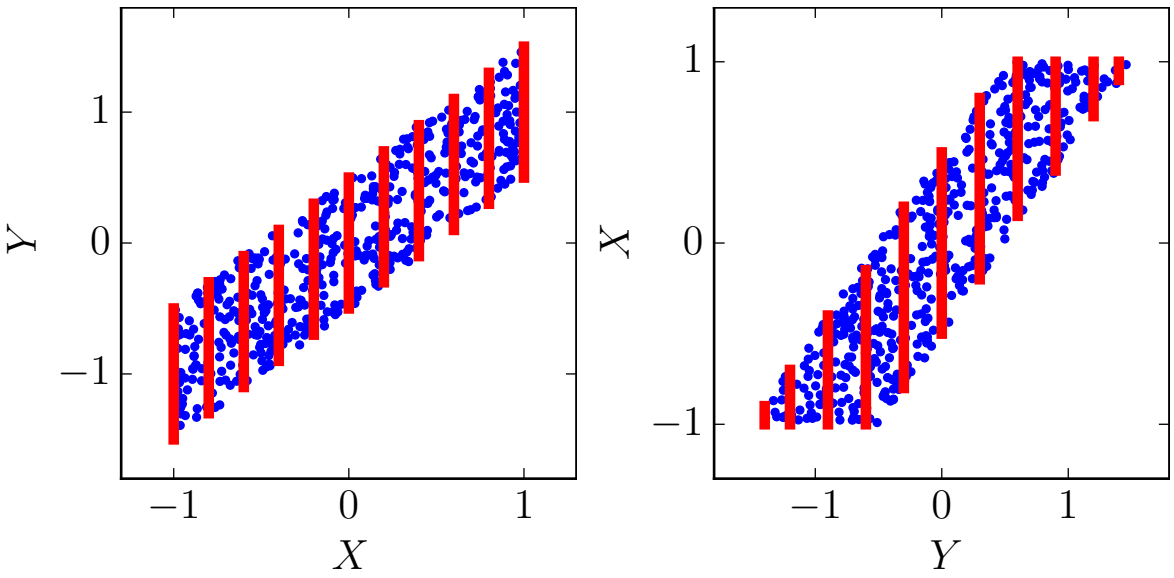
$$\begin{aligned} X &= N_1 \\ Y &= X + N_2, \end{aligned} \tag{2.3}$$

where N_1 and N_2 are two independent Gaussians, and $N_2 \sim \mathcal{N}(\mu, \sigma^2)$. Then trivially (X, Y) follows a bivariate Gaussian distribution. Given the intervention $\text{do}(X = 5)$, X becomes a point mass distribution and $Y \sim \mathcal{N}(\mu + 5, \sigma^2)$. The joint distribution arising from an intervention is thus different from the *observational* distribution. A fundamental property is that, despite the change in the joint distribution, the conditional $P(Y \mid X = x)$ remains the same before and after the invervention. Note, however, that the intervention $\text{do}(Y = 5)$ does not modify the distribution of X due to the asymmetrical assignment in the SEM. The ability to model interventions is a key property of causal models.

2.2 Observational Causal Discovery

Significant efforts in causality have focused on the problem of observational causal discovery. As discussed in the previous section, the gold standard for establishing causal relations is randomised experiments, which heavily rely on experimentation. In many real world settings, experiments are not feasible for ethical or practical reasons, yet discovering causal relations remains important.

Observational causal discovery (OCD) attempts to establish causal relations from purely observational data. Given a sample $\mathcal{D} \overset{iid}{\sim} P(X_1, \ldots, X_p)$, OCD methods try to recover the causal graph relating the random variables $X_1, \ldots, X_p$. In general, causation cannot be inferred from purely statistical dependence, and assumptions regarding the data generating process must be made for the causal graph to become identifiable from the observational distribution. We present methods for bivariate causal discovery, and leave out a large family of methods focusing on conditional independence properties which often assume that P is Markov, as well as faithfulness, see Peters et al. (2017, Chapter 6) for further details. The



(a) $Y = f(X) + \epsilon$ (b) $X = g(Y) + \eta$

Fig. 2.1 An Additive Noise Model (ANM).

application of additive noise models to more than two variables is discussed in Peters et al. (2014).

Bivariate causal discovery aims at differentiating the three scenarios in Principle 1 given a sample $\{(x_i, y_i)\}_{i=1}^n \overset{iid}{\sim} P(X, Y)$, i.e., we want to decide between $X \to Y$, $Y \to X$ or $X \leftarrow Z \to Y$ for some confounding variable Z. We present two families of methods for this task. They both rely on different assumptions which lead to a measurable causal footprint in the joint distribution.

2.2.1 Additive Noise Models

Additive noise models (Hoyer et al., 2009; Peters et al., 2014), or ANM, assume that in the correct causal direction, the effect is computed as a function of the cause with added independent noise. If we assume that X is a cause of Y, the ANM assumption implies that there exists a function f such that

$$Y = f(X) + \epsilon,$$

where $\epsilon \perp\!\!\!\perp X$. The ANM assumption leads to the identifiability of the causal direction if the converse does not hold, that is, there exists no function g such that $X = g(Y) + \eta$ with $\eta \perp\!\!\!\perp Y$. Measuring a dependence between the noise term η and the input Y in the anti-causal direction is the causal *footprint* which renders the causal relation *identifiable* from the joint distribution $P(X, Y)$. An example of an identifiable ANM is provided in Figure 2.1. Note

that additive noise models in the bivariate setting aim to distinguish $X \to Y$ from $Y \to X$ and do not consider the confounded case.

Example of non-identifiability The follow example is borrowed from Peters et al. (2014). Consider that the relation between X and Y is linear:

$$Y = \beta X + \epsilon, \tag{2.4}$$

where $\epsilon \perp\!\!\!\perp X$ and both X and ϵ are Gaussian. Then

$$X = \widetilde{\beta} Y + \widetilde{\epsilon},$$

where $\widetilde{\epsilon} \perp\!\!\!\perp Y$, $\widetilde{\beta} = \frac{\beta \mathrm{Var}(X)}{\beta^2 \mathrm{Var}(X) + \mathrm{Var}(\epsilon)}$ and $\widetilde{\epsilon} = X - \widehat{\beta} Y$. The existence of a mapping with independent noise terms in both directions makes linear relations with Gaussian noise non-identifiable. Nonetheless, Peters et al. (2014) show that (2.4) becomes identifiable if the noise variable ϵ is non-Gaussian, meaning that there exists no $(\widetilde{\beta}, \widetilde{\epsilon})$ such that $X = \widetilde{\beta} Y + \widetilde{\epsilon}$ with $\widetilde{\epsilon} \perp\!\!\!\perp Y$. Moreover, they provide further conditions for the identifiability of additive noise models in the non-linear setting.

2.2.2 Training Based Methods

Often, the ANM assumptions are violated. For instance, the noise may be multiplicative or heteroskedastic. Methods such as the randomised Causation Coefficient (RCC) (Lopez-Paz et al., 2015) proceed by first training a classifier to predict the causal direction given as input a sample from the joint distribution. RCC assumes that a training set $\mathcal{D} = \{(S_i, l_i)\}_{i=1}^n$ is available, where $S_i = \{(x_j^i, y_j^i)\}_{j=1}^{n_i} \overset{iid}{\sim} P^i(X_i, Y_i)$ is a sample from a distribution $P^i(X_i, Y_i)$, $l_i = 1$ if $X_i \to Y_i$ and $l_i = -1$ if $Y_i \to X_i$. These training pairs can be simulated by generating generic causal pairs or using real data. RCC creates a feature representation of each sample S_i using kernel mean embeddings (Smola et al., 2007), and learns a binary classifier using $\mathcal{D}$ to identify the causal footprints necessary to classify samples from new distributions.

At test time, a sample $\{(x_i, y_i)\}_{i=1}^n \overset{iid}{\sim} P(X, Y)$ is given to RCC for prediction. One may add a third class to the training data to allow for the confounded case. Similar methods include the Neural Causation Coefficient (Lopez-Paz et al., 2017), which computes features for S_i using a neural network instead of kernel mean embeddings.

2.3 Causal Inference Using Invariant Predictions

The previous sections presented two families of methods and assumptions which make the DAG identifiable from observational data. This section presents the approach from Peters et al. (2016) which relies on invariance properties verified by causal variables when experimental

data from several "tasks" is available. The ideas presented are related to the assumption of independence of cause and mechanisms (Janzing and Schölkopf, 2010; Schölkopf et al., 2012) and exogeneity (Zhang et al., 2015). These ideas motivate the approach for handling distribution shift presented in Chapter 3.

Let $Y \in \mathbb{R}$ be a variable we wish to predict from features $\mathbf{X} = (X^1, \ldots, X^p)$. Moreover, D different *tasks*, denoted as *environments* in Peters et al. (2016), are available at training time and we denote by $(\mathbf{X}^k, Y^k)$ the feature and target variables in task $k \in \{1, \ldots, D\}$. In the context of causal inference, these tasks may correspond to different interventions on the p features (but not on the target variable), or to different experimental settings in which the data was collected. We focus on linear SEMs with additive noise, but Peters et al. (2016) present an extension to the non-linear case. Peters et al. (2016) assume that the D tasks are related as follows:

Assumption 5 *There exists a vector $\gamma^* \in \mathbb{R}^p$ of linear causal coefficients such that:*

$$\forall k \in \{1, \ldots, D\},\ Y^k = \mu + (\gamma^*)^t \mathbf{X}^k + \epsilon,\ \ \epsilon \sim \mathbb{P}_\epsilon, \tag{2.5}$$

where ϵ has zero mean and finite variance. Let $S = \{u : \gamma_u^* \neq 0\}$ and assume that $\mathbf{X}_S$ is independent of ϵ.

The variables $\mathbf{X}_S$ are called causal predictors. Assuming that the underlying model is truly linear, what are the implications of Assumption 5 and what are its consequences for causal inference? Equation (2.5) is assumed to hold for all D tasks. In particular, the distribution of ϵ does not depend on the task, thus the conditional distribution $Y^k \,|\, \mathbf{X}_S^k$ is invariant across all tasks. The invariance under interventions of SEM relates to the notion of autonomy (Aldrich, 1989) or stability (Pearl, 2009), which state that modifying one equation in the SEM does not affect the remaining equations, or in other words that the equations in a SEM are autonomous. This assumption is realistic if the equations in the SEM are not seen as an algebraic equality but as mechanisms which link a set of causes (the parents in the corresponding DAG) to the value of the corresponding effect variable.

Peters et al. (2016) analyses the question of identifiability of the true causal parents of a target variable Y given multiple tasks for which Assumption 5 holds. In particular, given a set of interventions $k \in \{1, \ldots, D\}$ they define the set of identifiable causal predictors, which is proven to be a subset of the true causal predictors as defined in Assumption 5. Moreover, the number of identifiable coefficients grows as D increases, that is, observing more tasks leads to better estimation of the true causal parents. For linear SEMs with Gaussian noise, Theorem 2 in Peters et al. (2016) provides sufficient conditions for the identifiability of the true causal predictors. They also propose two algorithmic approaches to build confidence intervals on the set of identifiable causal predictors.

2.4 Causality in Machine Learning

At first sight, it is unclear how causality can be useful in standard machine learning. Standard learning theory assumes that in a supervised learning problem, the training and test data are i.i.d. draws from the same distribution $P(\mathbf{X}, Y)$ over features and target. This testing setting does not require knowledge of the causal structure relating the variables under consideration, as statistical dependence is sufficient for learning the optimal predictor in the population case.

We claim that causality can be useful for machine learning when discussing non i.i.d. problems, such as transfer learning and reinforcement learning, or when interpretability of the features is a requirement. This thesis presents in Chapter 3 a method for distribution shift inspired by invariance properties introduced in this chapter.

Reinforcement learning provides an intriguing test bed for causal assumptions. It is well known that agents trained to play games using model free reinforcement learning do not generalise when something as simple as the background of the game changes (Diuk et al., 2008; Kansky et al., 2017). In part, this is probably partly due to a training protocol which relies only on statistical dependencies to learn policies, and not on learning invariants of the environment. A framework for reinforcement learning inspired by causality would rather encourage learning a model of the world and understanding which modules are re-usable in different scenarios. For a robot interacting with the world, learning invariant properties of how objects react to physical laws and interactions with the robot should be of high importance for generalising to new environments.

Causality can also contribute to problems where understanding decisions and features is important. Often, we are not only interested in a prediction or a decision, but in *why* this decision was made. This can be due to a variety of reasons, including regulatory constraints, moral requirements or simply a desire for deeper understanding of the systems under study. Causality has also been applied as a tool for thinking about fairness and discrimination in automatic decision systems (Kilbertus et al., 2017).

More broadly, understanding and discovering causal laws is a key element in the development of machine intelligence and reasoning.

Chapter 3

Invariant Causal Representations for Transfer Under Distribution Shift

Standard machine learning assumes that the training and test data are drawn from the same distribution. Problems in which the test distribution differs are nonetheless ubiquitous in practice, a phenomenon we refer to as distribution shift. Building a music recommendation algorithm uses large amounts of data from a database of users, but not from a user new to the service, whose taste is unique. Data may also be gathered using different instruments or sources. In an image classification problem, the new test images may have been clicked with a new camera. In language, an algorithm may be trained with large amounts of text data coming from a large source such as Wikipedia, different from the language that will be given as input by the end users. The context of the examples may also shift: animals in an image training set may be depicted mostly in broad daylight, while the test set contains shots taken in the evening. Algorithms for distribution shift aim to address this mismatch between training and test distribution. A formal statement, as well as some of the standard methods and assumptions for distribution shift, are introduced in Section 3.1.

The goal of this chapter is to build representations of the data which transfer when the distributions shift. We introduce a new assumption for two specific modalities of domain shift of increasing difficulty: multi-task learning, where data from the test task are observed (Section 3.3), and domain generalisation, where no data from the test task are available (Section 3.2). This terminology is arguably not standard, but will be kept consistent throughout this chapter. In Baxter (2000) and several references therein, domain generalisation is referred to as learning to learn. For transfer to be possible, assumptions on how the training and test distributions are related must be made. We motivate that representations of the data for these challenging settings can be built on top of assumptions

motivated by causality. In particular, the notions of causal graph and *intervention* introduced in Chapter 2 may be used to relate the training and test tasks: the same causal graph may describe the generative process of the data in the different tasks, and each task corresponds to different set of interventions on this graph. Under these assumptions, we show that the feature mapping which simply selects a subset of causal features has attractive properties for domain generalisation and multi-task learning. Importantly, these transferable representations may have arbitrarily different distributions between tasks, yet the function or *mechanism* mapping them to the target variable is preserved, as motivated by Principle 2 in Chapter 2.

Finally, Section 3.4 focuses on a specific application of multi-task learning in which each task is computed by transforming a canonical distribution P via some *mechanisms*. In the motivating example, the canonical distribution P is the standard MNIST distribution, and the different mechanisms are simple transformations of these canonical digits such as translation or noise addition. The main motivation is that understanding how the new representations relate to the canonical distribution leads to re-usability: for instance, having trained a classifier on P using labelled data, we wish to use it on the transformed digits as well without gathering new labelled data. We provide an algorithm that learns approximate mappings from the transformed examples back to their canonical counterpart without labelled information. The transformed examples can be labelled by a classifier trained on the canonical distribution P. Importantly, we find empirically that the learned inverse mappings generalise to data from a new, unseen distribution, further enhancing their modularity and re-usability. Both our contributions in domain generalisation and multi-task learning motivate the idea that representations and models satisfying causal invariance properties are robust with respect to changes in the statistical properties of a probability distribution.

3.1 An Overview of Distribution Shift

3.1.1 From Standard Machine Learning to Distribution Shift

The standard framework for supervised machine learning assumes that a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n \overset{iid}{\sim} P(\mathbf{X}, Y)$ drawn i.i.d. from a probability distribution $P(\mathbf{X}, Y)$ on $\mathcal{X} \times \mathcal{Y}$ is available. The goal is to use $\mathcal{D}$ to build a model capable of predicting the *target* variable Y given *features* $\mathbf{X}$. Both regression and classification fall within this paradigm. Underlying this process lies the motivation that this model will perform well on new instances $\{\mathbf{x}_i\}_{i=1}^m \overset{iid}{\sim} P(\mathbf{X})$ for which the target variable needs to be predicted. In practice, it is most often assumed that the model will be tested on data drawn from the same distribution $P(\mathbf{X}, Y)$. This assumption is nonetheless strong and is often violated, which leads to a *distribution shift*.

Definition 6 (Distribution shift) *A learning problem suffers from* ***distribution shift*** *if the test data is drawn from a distribution P^T different from the training distribution P.*

Spam detectors, for instance, are trained on emails from users which receive attacks of a different nature than those experience by the end user. Even in problems typically associated with the i.i.d. paradigm such as image detection the risk of dataset shift exists: if the training images only contain dogs mostly resting in their bed, the model may perform arbitrarily bad given a dog playing outside.

We first present some key ideas from the standard i.i.d. prediction setting. We then introduce a framework for dataset shift introduced in Baxter (2000).

3.1.1.1 Overview of Standard Supervised Learning

The goal of standard supervised learning is to predict a target variable of interest $Y \in \mathcal{Y}$ given features $\mathbf{X} \in \mathcal{X}$. Let $P(\mathbf{X}, Y)$ be the corresponding joint distribution. In order to move forward, we require both a measure of the gap between the predictions and the target, and a specified set of functions to which we restrict our model. We measure how good our predictions are using a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$. Moreover, we restrict our model to a *hypothesis class* denoted $\mathcal{H}$. $\mathcal{H}$ may correspond, for example, to the set of linear models in $\mathbb{R}^d$, or the class of functions induced by a neural network with a fixed architecture.

During learning, our goal is to learn a model $h^* \in \mathcal{H}$ which minimises the expected loss or risk:

$$\mathcal{R}_P(h) = \mathbb{E}_{\mathbf{X},Y}\left[\ell\left(h(\mathbf{X}), Y\right)\right]. \tag{3.1}$$

In reality, we do not have access to the true data distribution P, but observe a sample $\{(\mathbf{x}_i, y_i)\}_{i=1}^n \overset{iid}{\sim} P(\mathbf{X}, Y)$. Empirical risk minimisation (Vapnik, 1992) minimises instead the empirical risk, defined as

$$\widehat{\mathcal{R}}_P(h) = \frac{1}{n}\sum_{i=1}^{n} \ell\left(h(\mathbf{x}_i), y_i\right).$$

A fundamental question which has interested practitioners and theorists alike is knowing how large the available sample from P must be to guarantee good performance, or *generalisation*, on *unseen data points* drawn i.i.d. from P. From a theoretical perspective, this question is well understood, and the convergence of the empirical risk to the true expected risk is governed by the VC dimension of the hypothesis class $\mathcal{H}$, see (Blumer et al., 1989; Vapnik, 1982) for convergence bounds and finite sample guarantees. The complexity of the hypothesis class $\mathcal{H}$ plays a fundamental role in machine learning, as there is a trade-off between between the bias and variance of the functions in the hypothesis class (Geman et al., 1992). Intuitively, if the function class allows for too much freedom, a small empirical error may be achieved on the training data at the expense of poor generalisation . Too poor a model class, however, leads to poor predictive performance.

Empirical risk minimisation does not deal with the common scenario in which the test data is drawn from a distribution which differs from the training distribution. The following section extends the previous setup to this important scenario.

3.1.1.2 Extension to the Multiple Task Problem

Let $\mathcal{P}$ be a set of joint probability distributions $P(\mathbf{X}, Y)$ on $\mathcal{X} \times \mathcal{Y}$ and $\mathcal{M}$ a distribution over $\mathcal{P}$.[1] $\mathcal{P}$ may consist on a finite ordered set of probability distributions $\mathcal{P} = (P^1, \dots, P^D)$, and $\mathcal{M}$ draws an index in $\{1, \dots, D\}$ according to some distribution and returns the corresponding P.

Moreover, define a family of hypothesis $\mathbb{H}$ such that each $\mathcal{H} \in \mathbb{H}$ is a set of functions $h : \mathcal{X} \to \mathcal{Y}$, similarly to the hypothesis classes in the previous section.

Definition 7 *We say that $P \in \mathcal{P}$ is a task[2], and $\mathcal{M}$ is a distribution over tasks.*

Given a non-negative loss function $\ell : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$, our goal is to find a family of hypothesis $\mathcal{H}^*$ which minimises the following expected risk over distributions drawn from $\mathcal{M}$:

$$\mathcal{R}_{\mathcal{M}}(\mathcal{H}) = \mathbb{E}_{P \sim \mathcal{M}} \left[\inf_{h \in \mathcal{H}} \mathcal{R}_P(h) \right]$$

This problem is referred to in Baxter (2000) as the *bias learning* problem, or *learning to learn* (Thrun and Pratt, 1998). The bias learner aims to find a hypothesis class, and not only a function, which contains a good solution for tasks drawn according to $\mathcal{M}$, and is closely related to meta-learning (see for instance Lemke et al. (2015)).

Similarly to empirical risk minimisation, we do not have access to the true distribution over tasks $\mathcal{M}$. Hence, we are given a sample of tasks $\{P^k\}_{k=1}^D \overset{iid}{\sim} \mathcal{M}$ and for each $k \in \{1, \dots, D\}$, a sample $\{(\mathbf{x}_i^k, y_i^k)\}_{i=1}^{n_k} \overset{iid}{\sim} P^k$. As a proxy of the true expected risk, one may minimise the empirical risk $\widehat{\mathcal{R}}_{\mathcal{M}}(\mathcal{H}) = \frac{1}{D} \sum_{k=1}^D \inf_{h \in \mathcal{H}} \widehat{\mathcal{R}}_{P^k}(h)$. Uniform convergence results of $\widehat{\mathcal{R}}_{\mathcal{M}}(\mathcal{H})$ to $\mathcal{R}_{\mathcal{M}}(\mathcal{H})$ are established in Baxter (2000).

Learning theory in the i.i.d. setting analyses how many examples drawn from P need to be seen during training in order for the solution $h^* \in \mathcal{H}$ to generalise well to unseen examples drawn from the same distribution P. In the multi-task setting, the main goal is no longer to perform well in one specific task, but rather in any task drawn from $\mathcal{M}$. The following question becomes central for transfer learning in general, and remains a major challenge for the machine learning community:

In the multiple task problem, how many tasks need to be observed in order to achieve good generalisation performance for unseen tasks? Under which assumptions can such transfer take place?

[1] See (Baxter, 2000) for a more rigorous mathematical description of $\mathcal{M}$.

[2] In this work, we use the term task and domain interchangeably. Many authors make a distinction between the two, see Pan and Yang (2010), but the distinction is not essential.

method	training data from	test domain
Domain generalisation (DG)	$(\mathbf{X}^1, Y^1), \ldots, (\mathbf{X}^D, Y^D)$	$T := D + 1$
	$(\mathbf{X}^1, Y^1), \ldots, (\mathbf{X}^D, Y^D), \widetilde{\mathbf{X}}^{D+1}$	
Multi-Task Learning (MTL)	$(\mathbf{X}^1, Y^1), \ldots, (\mathbf{X}^D, Y^D)$	T=1,. . . ,D
	$(\mathbf{X}^1, Y^1), \ldots, (\mathbf{X}^D, Y^D), \widetilde{\mathbf{X}}^1, \ldots, \widetilde{\mathbf{X}}^D$	

Table 3.1 Taxonomy for domain generalisation (DG) and multi-task learning (MTL). Each problem can either be used without (first line) or with (second line) additional unlabelled data.

Knowing which knowledge from other tasks can be used when observing a new task seems particularly important. For instance, if all the tasks are completely unrelated, knowledge transfer can be expected to fail, or even hurt performance, see for instance (Ben-David et al., 2010; Rosenstein et al., 2005). As an illustration, consider standard image classification in which the goal is to predict an image label from pixel features. If the tasks consist on a large variety of images, taken from diverse cameras and in a representative set of backgrounds, the availability of multiple tasks should make it easier to find features which are predictive of a given label, and not simply correlated *in a given dataset*. On the other extreme, consider a binary classification dataset drawn from an arbitrary distribution P, and consider the distribution Q which has identical support to P but simply flips the labels. While each task may be easy to solve independently, treating them together makes the problem significantly harder, as it gets close to random guessing.

Setup. During training, we observe D probability distributions $P^1, \ldots, P^D$ referred to as *training tasks*. Moreover, a sample $\{(\mathbf{x}_i^k, y_i^k)\} \overset{iid}{\sim} P^k$ from each training task is available. We define two different transfer learning problems depending on the nature of the test distribution P^T. Domain generalisation (DG) models are tested on a dataset drawn from a new distribution $P^T = P^{D+1}$ which is not included in the training tasks. DG is a challenging problem and assumptions regarding the similarity between tasks must be made for learning to be possible. Multi-task learning (MTL) tests on unseen data drawn from one or several of the training tasks. In this case, the goal is to exploit similarities between the training tasks and perform better than learning each task independently. Both DG and MTL can get access to unlabelled data from the test task during training. Table 3.1 presents a summary taxonomy of DG and MTL.

The field of transfer learning considers a wider range of settings we do not discuss. These include, among others, domain adaptation and learning to learn, see Pan and Yang (2010) and references therein for further details.

We have mentioned that the distributions change between training and test, but have not presented methods for measuring the difference between distributions given data. The

following section provides tools for measuring distances between probability distributions. While certainly not exhaustive, the presented methods are essential components of the DG procedures introduced in Section 3.2.

3.1.2 Measuring Differences Between Distributions

We are studying problems in which *distinct probability distributions* are available. Since in practice we do not have access to the probability distributions themselves, we require a method to assess the distance between two probability distributions P and Q based on two samples $\{x_i\}_{i=1}^n \overset{iid}{\sim} P$ and $\{y_i\}_{i=1}^m \overset{iid}{\sim} Q$. In particular, we wish to perform a *two sample test*, which tests the null hypothesis $H_0 : P = Q$ given the two samples. First, we introduce Levene's test for equality of variances (Levene, 1960), which tests for equality of the variances of several distributions given only samples. We then introduce the Maximum Mean Discrepancy (MMD) and a kernel two sample test. Finally, we propose a novel method for testing whether D scalar samples are drawn from the same distribution, which becomes a key component of the method introduced in Section 3.2. This is certainly not an exhaustive list, and such tests are ubiquitous in machine learning. Two sample tests are particularly relevant for adversarial generative models (Goodfellow et al., 2014), as the generative component of the network is trying to fool a discriminator, which is essentially performing a two sample test between the real data and the fake data generated by the model. This idea was already used for estimating unnormalised parametric models with Noise Contrastive Estimation (Gutmann and Hyvärinen, 2010). More recently, Lopez-Paz and Oquab (2016) introduce a two sample test by training a binary classifier to discriminate between the two distributions.

3.1.2.1 Levene's Test For Equality of Variances

Let P and Q be two real valued probability distributions. Two samples $\{x_i\}_{i=1}^n \overset{iid}{\sim} P$ and $\{y_i\}_{i=1}^m \overset{iid}{\sim} Q$ are available. Let $\overline{x} = \frac{1}{n}\sum_{i=1}^n x_i$ and $\overline{y} = \frac{1}{m}\sum_{i=1}^m y_i$ be the empirical mean of both samples. Let $u_i = |x_i - \overline{x}|$ and $v_i = |y_i - \overline{y}|$, and denote by $\overline{u}$ and $\overline{v}$ for the corresponding averages. Finally, denote by $\overline{z}$ the overall average $\frac{1}{n+m}\left(\sum_{i=1}^n u_i + \sum_{j=1}^m v_i\right)$.

We define the test statistic z as

$$z = (n + m - 2)\frac{n(\overline{u} - \overline{z}) + m(\overline{v} - \overline{z})}{\sum_{i=1}^n (u_i - \overline{u}) + \sum_{i=1}^m (v_i - \overline{v})},$$

and we test the null hypothesis $H_0 : \mathrm{Var}(P) = \mathrm{Var}(Q)$. Under H_0, z follows an F-distribution $F(1, n + m - 2)$. The test rejects H_0 at level α if $z > F_{\alpha,1,n+m-2}$, where $F_{\alpha,1,n+m-2}$ is the upper critical value of the corresponding F distribution at level α.

For our purposes, Levene's test is relevant when both distributions are Gaussian. If we assume they both have zero mean, their variance completely determines the distribution, and the test can be used as a two sample test. The test introduced in Levene (1960) is more

general, as it can be applied to test the variances of the variance of D distributions. We restricted the exposition to two distributions for clarity.

3.1.2.2 Kernel Two Sample Tests

Kernel methods provide an intuitive, theoretically grounded approach for non-linear machine learning (Schölkopf et al., 2002). One of the most appealing properties of kernel methods is that they render some linear methods non-linear by use of the *kernel trick*. A function $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a positive definite kernel if for all $n \in \mathbb{N}$, for all $x_1, \ldots, x_n \in \mathcal{X}$ and $c_1, \ldots, c_n \in \mathbb{R}$, $\sum_{i=1}^{n}\sum_{j=1}^{n} c_i c_j \kappa(x_i, x_j) \geq 0$. A positive definite kernel is associated to a unique Hilbert space $\mathcal{H}$ called Reproducing Kernel Hilbert Space (RKHS) with dot product $\langle ., . \rangle$. There exists a feature map $\phi : \mathcal{X} \to \mathcal{H}$ such that for any two $x, y \in \mathcal{X}$, $\langle \phi(x), \phi(y) \rangle = \kappa(x, y)$. The previous property lies at the heart of kernel methods: the feature representation of a data point $x \in \mathcal{X}$ implied by the kernel κ is $\phi(x)$, which lives in a potentially infinite dimensional space. If the algorithm only depends on dot products in the input space, it can often be written in terms of the dot product in the RKHS, and the feature maps do not need to be computed explicitly. This convenient trick is denoted the kernel trick and is at the core of techniques such as non-linear support vector machines (e.g., Schölkopf et al. (2002)).

Recent efforts aim to extend the embedding of data *points* $x \in \mathcal{X}$ to an RKHS to embed whole probability distributions P, see Sriperumbudur et al. (2010) and references therein. Given a probability distribution P and a positive definite kernel κ, we define the *mean embedding* of P as

$$\mu_\kappa(P) = \mathbb{E}_{X \sim P} \left[\phi(X) \right].$$

$\mu_\kappa(P)$ is such that for any function $f \in \mathcal{H}$, $\langle f, \mu_\kappa(P) \rangle = \mathbb{E}_{X \sim P} \left[f(X) \right]$, so computing expectations with respect to P becomes a linear operation in the RKHS.

But how can we use these mean embeddings to measure the distance between distributions? Gretton et al. (2012) propose a two sample test based on this idea. If a kernel κ is *characteristic* (such as the square exponential kernel with length scale σ: $\kappa(x, y) = \exp\left(\|x - y\|^2 / 2\sigma^2\right)$), the mean embedding is an injective mapping (Sriperumbudur et al., 2010). In other words, given two probability distributions P and Q, $\mu_\kappa(P) = \mu_\kappa(Q)$ if and only if $P = Q$.

Define the Maximum Mean Discrepancy (MMD) of distributions P and Q given a set of functions $\mathcal{F}$ as $\text{MMD}(\mathcal{F}, P, Q) = \sup_{f \in \mathcal{F}} \left(\mathbb{E}_{X \sim P} \left[f(X) \right] - \mathbb{E}_{Y \sim Q} \left[f(Y) \right] \right)$. Gretton et al. (2012) prove that

$$\text{MMD}(U(\mathcal{H}), P, Q)^2 = \|\mu(P) - \mu(Q)\|_{\mathcal{H}}^2 , \tag{3.2}$$

where $U(\mathcal{H})$ is the unit ball of $\mathcal{H}$ and $\|.\|$ is the dot product norm in the $\mathcal{H}$. Given the injectivity of the mean embedding, performing a two sample test is equivalent to testing

$$H_0 : \text{MMD}(U(\mathcal{H}), P, Q)^2 = 0$$

when a characteristic kernel is used.

If we are given two i.i.d. samples $\{x_i\}_{i=1}^n$ and $\{y_i\}_{i=1}^m$ from P and Q respectively, Smola et al. (2007) prove that

$$\begin{aligned}\widehat{\mathrm{MMD}}(U(\mathcal{H}), P, Q)^2 = &\frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{j \neq i} \kappa(x_i, x_j) + \frac{1}{m(m-1)} \sum_{i=1}^{m} \sum_{j \neq i} \kappa(y_i, y_j) \\ &- \frac{2}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} \kappa(x_i, y_j)\end{aligned} \tag{3.3}$$

is an unbiased estimator of (3.2), which Gretton et al. (2012) use to construct a two sample test.

Kernel two sample tests can be directly extended to test independence (Gretton et al., 2005) by estimating the Hilbert Schmidt Independence Criterion (HSIC). HSIC applied ideas closely related to the previous two sample test by embedding the joint distribution $P(X, Y)$ and the product of the marginals $P(X)P(Y)$. Given a characteristic kernel, the null hypothesis $H_0 : X \perp\!\!\!\perp Y$ is equivalent to $H_0 : \mathrm{MMD}(U(\mathcal{H}), P(X, Y), P(X)P(Y))^2 = 0$, which can be estimated using the unbiased estimator in (3.3).

3.1.2.3 Testing Whether D Samples are Drawn From the Same Distribution

We introduce a method for testing whether D samples are drawn from the same scalar distribution. More precisely, we want to test the null hypothesis

$$H_0 : \forall k, k' \in \{1, \ldots, D\},\ P^k = P^{k'}$$

given D scalar samples $\{x_i^k\}_{i=1}^{n_k} \sim P^k$ for $k \in \{1, \ldots, D\}$.

We propose to test this null hypothesis by introducing a discrete random variable K with values in $\{1, \ldots, D\}$ whose outcome indicates the index of the task corresponding to a given data point. If all the examples are drawn from the same probability distribution, K is independent of the data generating random variable X. This test is a direct application of HSIC (Gretton et al., 2007) but to our knowledge, is novel.

Let $K \sim P(K)$ be a discrete random variable over $\{1, \ldots, D\}$ and let X be a scalar random variable such that $P(X, K) = P(X \mid K)P(K)$ with $P(X \mid K = k) = P^k$. The outcome of K corresponds to the index of the distribution from which X is sampled. It is straightforward to see that, if X and K are independent, $P(X \mid K = k) = P(X)$, thus for all $k \in \{1, \ldots, D\}$, $P^k = P(X)$ and all the distributions are equal.

We pool all the data from the different distributions into a sample $Z = (x_i, k_i)_{i=1}^n$ drawn from $P(X, K)$, and test for independence of X and K using HSIC. Two characteristic kernels are used: a kernel κ for embedding the data and a trivial kernel d such that $d(i, j) = \delta_{ij}$ for K. Let therefore $\mathrm{HSIC}(Z)$ denote the value of the HSIC between X and K, and let

$\text{HSIC}_b(Z)$ be the corresponding test statistic. Then H_0 is accepted if if leads to accepting the null hypothesis of independence between X and K at level δ.

3.1.3 Assumption For Distribution Shift

In this section, we discuss some of the assumptions and methods which make transfer learning possible, both in DG and MTL. The list is certainly not exhaustive, and more detailed surveys can be found in Pan and Yang (2010).

3.1.3.1 Covariate Shift

An assumption which has been widely used for DS is covariate shift (CS). CS assumes that the distribution of the covariates $\mathbf{X}$ changes between tasks, but the conditional mapping $P(Y \mid \mathbf{X})$ remains unchanged.

Definition 8 (Covariate Shift) *We say that covariate shift holds for tasks $P^1, \ldots, P^k, P^T$ if*

$$\forall k \in \{1, \ldots, D\}, \;\; P^k(Y \mid \mathbf{X}^k) = P^T(Y \mid \mathbf{X}^T). \tag{3.4}$$

The joint distribution of features and target in task k can be factorised as $P^k(\mathbf{X}, Y) = P^k(\mathbf{X})P^k(Y \mid \mathbf{X}) = P^k(\mathbf{X})P^T(Y \mid \mathbf{X})$. Therefore, covariate shift imposes important constraints on how the joint distribution differs between tasks, as it can only vary through the marginal distribution of the features $P^k(\mathbf{X})$. Covariate shift arises naturally in many problems and is attractive given its simplicity. An example in which CS could be applied is spam detection. A spam classifier can be trained on emails from a large amount of users. At deployment time, however, the classifier should perform well on emails from a new user for which very few data points are observed. The distribution of emails received by the user is very likely different from the training distribution, and this must be taken into account for prediction.

We present two ways in which CS can be exploited in practice. This list is certainly not exhaustive.

First, we can use importance sampling (Hammersley and Handscomb, 1965) to estimate the expected loss in the test task when only an unlabelled sample is given. This problem is common in domain adaptation, as a small labelled dataset may be available together with a larger unlabelled dataset from the test distribution. One such example may be a small labelled image classification dataset taken from a specific source, and a large dataset of images crawled from the web without labels.

Consider a setting with one source task P^1 and one test task P^T. A labelled sample $\{(\mathbf{x}_i^1, y_i^1)\}_{i=1}^{n_1} \sim P^1$ and and unlabelled sample $\{\mathbf{x}_i^T\}_{i=1}^{n_T}$ are available. This setting is very common in practice, as large amounts of data may be available, but only a few labelled examples are available from a source distribution. Given the invariance of the mapping from

$\mathbf{X}$ to Y, importance sampling aims to re-weight the examples in the test task in order to make them more representative of the source task, for which labels are available. When training to perform well on the test task, our goal is to minimise the expected loss on the test task $\mathcal{R}_{P^T}$ defined in (3.1) in some predefined hypothesis class $\mathcal{H}$.

$$\begin{aligned}
\mathcal{R}_{P^T}(h) &= \mathbb{E}_{(\mathbf{X},Y)\sim P^T}\left(\ell(h(\mathbf{X}),Y)\right) \\
&= \int_{\mathcal{X}\times\mathcal{Y}} \ell(h(\mathbf{x}),y)p^T(\mathbf{x},y)dxdy \\
&= \int_{\mathcal{X}\times\mathcal{Y}} \ell(h(\mathbf{x}),y)\frac{p^T(\mathbf{x},y)}{p^1(\mathbf{x},y)}p^1(\mathbf{x},y)dxdy \\
&= \mathbb{E}_{(\mathbf{X},Y)\sim P^1}\left(\frac{p^T(\mathbf{X},Y)}{p^1(\mathbf{X},Y)}\ell(h(\mathbf{X}),Y)\right) \\
&= \mathbb{E}_{(\mathbf{X},Y)\sim P^1}\left(\frac{p^T(\mathbf{X})}{p^1(\mathbf{X})}\ell(h(\mathbf{X}),Y)\right).
\end{aligned}$$

The simplification in the last line results from Definition 8. The idea behind importance sampling is hence to replace the computation of the expected loss with respect to an unavailable distribution for the expectation of $\frac{p^T(\mathbf{X})}{p^1(\mathbf{X})}\ell\left(h(\mathbf{X}),Y\right)$ with respect to the source distribution P^1. The main caveat is that we generally don't know $\frac{p^T}{p^1}$. The naive approach to this problem consists on doing density estimation of p^T and p^1 separately, and computing the ratio for each example (for instance, using kernel density estimation, see Scott (2015)). However, density estimation techniques are known to be difficult as the dimensionality of the data increases (Bickel et al., 2007; Huang et al., 2007; Quionero-Candela et al., 2009; Shimodaira, 2000; Sugiyama et al., 2008).

CS may also be exploited in a regression setting as it justified pooling the data. Given samples $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n_k}$ for $k \in \{1,\ldots,D\}$, one may pool the data in order to estimate the conditional mean $f(x) = \mathbb{E}(Y^T \mid \mathbf{X}^T = x)$. This increases the amount of available data, and thus should lead to a better estimate of the population mean.

Both importance sampling and pooling suffer from estimation problems. Consider the problem of learning a regression function $\hat{f}$ as in the pooling example. It remains unclear weather the estimated regression function $\hat{f}$ has good predictive power in data from the test task if it is not among the training tasks. Indeed, if $P^T(\mathbf{X})$ has probability mass in regions not overlapping with the training tasks, no guarantees can be established. This problem is not present for some simple model classes, such as linear models. It is important to note that this is an estimation problem only, not a theoretical one, since domain extrapolation is a difficult statistical problem. The same issue is present for importance sampling, since the re-weighting method presented assumes that the distributions share the same support.

3.1.3.2 Marginalised De-noising Autoencoders for Domain Adaptation

Chen et al. (2012) propose a method for distribution shift based on learning an autoencoding of the data. In short, the goal of an autoencoder is to find a low-dimensional feature representation of the data which achieves high reconstruction accuracy. In this regard, its role is analogous to compression on the features, the underlying assumption being that the original features are correlated, and the data features live in a lower dimensional space. Similar assumptions are common in other transfer learning methods. A generic autoencoder contains two components: an encoder function $e : \mathbb{R}^d \to \mathbb{R}^{d'}$ with $d' < d$ and a decoder function $d : \mathbb{R}^{d'} \mapsto \mathbb{R}^d$. The encoder e computes a low dimensional representation of the input $\mathbf{x}$, and d maps the compressed feature vector back to $\mathbf{x}$. A standard autoencoder is thus trained to minimise the expected reconstruction loss $\mathbb{E}_{\mathbf{x}}\left[(\mathbf{x} - d(e(\mathbf{x}))^2\right]$.

De-noising autoencoders (Glorot et al., 2011) (SDA) perform a similar task in the context of transfer learning. Given a sample from the source distribution $\{\mathbf{x}_i^1\}_{i=1}^n \sim P^1(\mathbf{X})$ and the test distribution $\{\mathbf{x}_i^T\}_{i=1}^m \sim P^T$, SDA corrupts the data in both samples with masking noise and aims to reconstruct the un-corrupted input, where the encoder e and the decoder d are parametrised by neural networks. While Glorot et al. (2011) perform stochastic gradient descent to optimise the reconstruction loss, Chen et al. (2012) re-write the objective by marginalising out the masking noise, leading to a closed form expression for the features. This results in Marginalised De-noising Autoencoders for Domain Adaptation (mSDA).

The authors in Chen et al. (2012) motivate the success of mSDA with an example from sentiment analysis. Assume that the test domain consists on reviews for books, and the source domain reviews for cycling equipment. The word 'gripping' is relevant for classifying book reviews in the test domain, but this word is rare in the context of cycling equipment. By adding masking noise, the model should learn to use correlated features such as 'good' to reconstruct these domain specific words. SDA are extended to several layers in order to encourage more flexible encoders and decoders, and can be generalised to more than two source distributions.

3.1.3.3 Domain-Invariant Component Analysis

Muandet et al. (2013) propose Domain-Invariant Component Analysis (DICA) to address the problem of DG. Similarly to the formalism introduced in Section 3.1.1.2, we denote by $\mathcal{P}$ a family of joint distributions $P(\mathbf{X}, Y)$ over $\mathcal{X} \times \mathcal{Y}$. Each element of $\mathcal{P}$ is called a task. Let $\mathcal{M}$ be a distribution over $\mathcal{P}$. D training tasks $P^1, \ldots, P^D$ sampled from $\mathcal{M}$ are available, as well as samples $\{(\mathbf{x}_i^k, y_i^k)\}_{i=1}^{n_k}$ for $k \in \{1, \ldots, D\}$. We denote by $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ the pooled sample, where $n = \sum_{k=1}^D n_k$.

DICA aims to learn a transformation $h : \mathcal{X} \to \mathcal{Y}$ achieving the following:

i) the empirical distance between the distribution of the transformed features $\{h(\mathbf{x}_i^k)\}_{i=1}^{n_k}$, for $k \in \{1, \ldots, D\}$ is small.

ii) there is no loss of information regarding the conditional mapping from $\mathbf{X}$ to Y, such that $Y \perp\!\!\!\perp \mathbf{X} \,|\, h(\mathbf{X})$.

In order to quantify the first condition, we require a distance measure between the distribution of the transformed features $h(\mathbf{X})$. Let $P^k(\mathbf{X})$ denote the marginal distribution of the features in task k for $k \in \{1, \ldots, D\}$. As a measure of distance between the distributions of $h(\mathbf{X}^k)$, the authors propose to use distributional variance. For other measures of the distance between probability distributions, see Section 3.1.2.

Definition 9 (Distributional Variance) *Let $\overline{P}_{\mathbf{X}} = \frac{1}{N}\sum_{k=1}^{D} P^k(\mathbf{X})$ be the uniform mixture of the marginals in the source distributions. Let $\mathcal{M}$ be a uniform distribution over the source marginals. Let κ be a characteristic kernel on $\mathcal{X}$ and $\mathcal{H}$ the corresponding RKHS.*

The distributional variance of $\mathcal{M}$ is

$$\mathcal{V}_{\mathcal{H}}(\mathcal{M}) = \frac{1}{N}\sum_{k=1}^{D} \|\mu_\kappa(P^k(\mathbf{X})) - \mu_\kappa(\overline{P}_{\mathbf{X}})\|^2. \tag{3.5}$$

Moreover, Theorem 1 in Muandet et al. (2013) states that the distributional variance is zero if and only if $P^1 = \ldots = P^D$. Theorem 2 in Muandet et al. (2013) introduces a consistent estimator $\hat{\mathcal{V}}_{\mathcal{H}}(\{x_i\}_{i=1}^n)$ of $\mathcal{V}_{\mathcal{H}}(\mathcal{M})$ given a pooled sample $\{\mathbf{x}_i\}_{i=1}^n$ drawn from tasks in $\mathcal{P}$. DICA learns a transformation of the data features for which the distributional variance of the transformed features is small.

3.1.3.4 Multi-task Feature Learning

MTL can be tackled by learning low-dimensional features which are shared across the different tasks. Given samples from D tasks, we wish to jointly learn D regression functions $f^k : \mathcal{X} \to \mathcal{Y}$ for the different tasks. As usual in MTL, the goal is to outperform learning all the tasks independently. In Argyriou et al. (2007), the authors make the following assumption:

Assumption 10 (Multi-task Feature Learning) *The D regression functions are a weighted average of J feature representations:*

$$\forall k \in \{1, \ldots, D\},\ \forall \mathbf{x} \in \mathcal{X},\ f^k(x) = \sum_{j=1}^{J} a_{jk} h^j(\mathbf{x}).$$

Moreover, the weight of most features is zero across all the tasks, that is, only the same (small) set of features is used in all tasks.

The feature functions h^j in Argyriou et al. (2007) are linear and $\mathcal{X} = \mathbb{R}^d$. Thus, the feature representations can be written as $h^j(\mathbf{x}) = \langle u^j, \mathbf{x}\rangle$ where the vectors u^j are orthonormal and $\langle .,. \rangle$ denotes the usual dot product in $\mathbb{R}^d$. This implies in turn the linearity of the regression functions f^k :

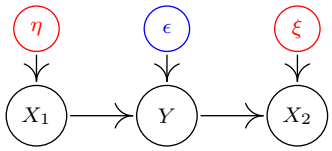


Fig. 3.1 Example of a causal DAG $\mathcal{G}$.

$$\forall k \in \{1, \ldots, D\},\ f^k(\mathbf{x}) = \langle w^k, \mathbf{x} \rangle,$$

where $w^k = \sum_j a_{jk} u^j$. Let A, W and U be the matrices constructed by stacking vectors u, a and w (therefore $W = UA$). The joint sparsity of the feature vectors implies that matrix W has low rank, as matrix A has many rows which are identically zero. The goal of the MTL algorithm is to learn the matrix of feature vectors U and the weights A for each of the tasks. The authors encourage sparsity across tasks via grouped L_1 regularisation:

$$E(A, U) = \sum_{k=1}^{D} \sum_{i=1}^{n_k} L(y_i^k, \langle a^k, U^T \mathbf{x}_i^k \rangle) + \lambda \|A\|_{2,1}^2. \tag{3.6}$$

Here, $\|.\|_{2,1}$ denotes de $(2,1)$-norm defined by $\|A\|_{2,1} = \sum_{j=1}^{d} \|a^j\|_2$ where a^j denotes the rows of A. The regularisation term corresponds to computing the L_2 norm of the rows of A and then computing the L_1 norm of the vector of those norms. The Lasso (Tibshirani, 1994) corresponds to L_1 regularisation in the single task problem, and is known to lead to sparse weight vectors. If we denote by $\mathbb{O}^d$ the set of orthogonal matrices in $\mathbb{R}^d$ (we assumed that the vectors u^j are orthogonal), the goal is to solve the following optimisation problem:

$$A^*, U^* = \{\min E(A, U),\ U \in \mathbb{O}^d,\ A \in \mathbb{R}^{d \times D}\} \tag{3.7}$$

The authors prove that (3.7) is equivalent to a convex optimisation problem and define an iterative algorithm to solve for A and U.

3.2 Invariant Models for Domain Generalisation

Motivation. We introduce a novel approach to DG based on invariant properties of causal models. Consider the graphical model in Figure 3.1, and that the goal is to predict Y given $\mathbf{X} = (X_1, X_2)$. At training time, we receive a sample $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ drawn i.i.d. from a distribution P, and we wish to learn a regression model which we can deploy on a dataset $\{(\widetilde{\mathbf{x}}_i, \widetilde{y}_i)\}_{i=1}^m$ drawn i.i.d. from some distribution $P^T \neq P$. Without any further regularity assumptions between P and P^T, this challenging DG problem cannot be tackled in a principled way. Nonetheless, we assume that P and P^T are both generated following the SEM depicted in Figure 3.1, where the distribution of the variables in red differs in both tasks, while the

distribution of ϵ remains the same. More specifically, we assume that

$$Y = \alpha X_1 + \epsilon,$$

where $\epsilon \perp\!\!\!\perp X_1$ and α is the same in both tasks. Moreover, we assume that

$$X_1 = \eta \qquad \text{and} \qquad X_2 = \gamma Y + \xi,$$

where η and ξ have different distributions in both tasks, while $\gamma \in \mathbb{R}$ remains fixed. How should one build a model that generalises to P^T?

One solution is to fit the least squares estimator $\beta^* = \arg\min_\beta \sum_{i=1}^n (y_i - \beta^t \mathbf{x}_i)^2$, and perform prediction in the test task using β^*. This solution works well if covariate shift holds, that is, $P(Y \mid \mathbf{X}) = P^T(Y \mid \mathbf{X})$ but the marginals $P(\mathbf{X})$ and $P^T(\mathbf{X})$ differ. A second possibility, following the observation that $P(Y \mid X_1) = P^T(Y \mid X_1)$, consists of using only X_1 as a predictor, resulting in the estimator $\beta^{\mathcal{G}} = (\alpha, 0)$. This approach is motivated by the idea of using a subset of predictors leading to invariant conditionals, see Section 2.3 in Chapter 2.

Given toy data generated from the DAG in Figure 3.1, we first compute the log mean squared error on held-out test data from distribution P. As expected, we obtain $\log(\text{MSE}(\beta^*)) = -2.94$ and $\log(\text{MSE}(\beta^{\mathcal{G}})) = -1.6$, that is, the least squares predictor outperforms $\beta^{\mathcal{G}}$. This is unsurprising, as β^* approximates the conditional mean $\mathbb{E}[Y \mid \mathbf{X}]$ which is optimal for i.i.d. prediction.

Assume now that we generate 1000 test tasks P^T by randomly sampling the variance of η and ξ, and compute the log mean squared error on data from these tasks both with β^* and $\beta^{\mathcal{G}}$, see Figure 3.2.

In these test tasks, β^* performs badly and leads to high variance in the errors, while $\beta^{\mathcal{G}}$ leads to small variance, and the errors are as expected close to $\mathbb{E}(\epsilon^2)$ in all the test tasks. This observation is central to the contributions in this section. We will assume that the source tasks and test tasks are related by the invariance of the conditional $Y \mid \mathbf{X}_{S^*}$ for some subset S^* of the covariates, corresponding to $S = \{1\}$ in the toy example. We show that under these assumptions, predicting using variables in S^* is optimal in an adversarial setting, which can intuitively be observed in Figure 3.2.

Properties of the invariance of mechanisms in causal systems motivate our approach to DG. We assume a relaxation of covariate shift, which we assume holds only for a subset of the features. In causal terminology, assuming invariance of conditionals makes sense if the conditionals represent causal mechanisms (e.g., Hoover, 1990). Intuitively, we expect that a causal mechanism is a property of the physical world, and it does not depend on what we feed into it. In a simplistic model, a variable such as "performance at the workplace" could be a function of the "years of quality education", and it is sensible to assume that this hold regardless of the country in which the data is measured. In these countries, however, the

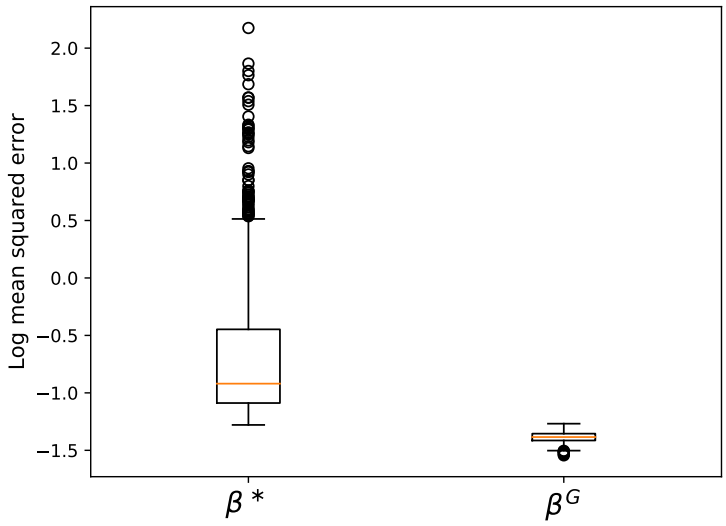


Fig. 3.2 Log mean squared errors of both predictors β^* and $\beta^{\mathcal{G}}$ on test tasks generated randomly from the graph in Figure 3.1. In each task, the variances of η and ξ were randomly sampled from an exponential distribution. The variance in the errors computed by predicting with β^* is large, while the errors incurred by $\beta^{\mathcal{G}}$ have small variance and are centered around $\log(\mathbb{E}(\epsilon^2))$.

distribution of years of quality education among the population might drastically differ. If the input, which in this case coincides with the covariates, shifts, the mechanism should thus remain invariant (Hoover, 1990; Janzing and Schölkopf, 2010; Peters et al., 2016). In the anticausal direction, however, a shift of the input usually leads to a changing conditional (Schölkopf et al., 2012), see for instance the toy experiment in Figure 3.2. In practice, prediction problems are often not causal — we should allow for the possibility that the set of predictors contains variables that are causal, anticausal, or confounded. We thus expect that there is a *subset* S^* of predictors, referred to as an **invariant set**, for which the covariate shift assumption holds true, i.e., the conditionals of output given predictor $Y^k \,|\, \mathbf{X}^k_{S^*}$ are invariant across $k \in \{1, \ldots, D, T\}$.

While not directly discussed in this section, the relation to covariate shift implies that once an invariant set is known, traditional methods for covariate shift can be applied as a black box to features in the subset, see Figure 3.3.

3.2.1 Assumption for Domain Generalisation

Consider a DG regression problem with source tasks $P^1, \ldots, P^D$, where $(\mathbf{X}^k, Y^k) \sim P^k$ for $k \in \{1, \ldots, D\}$.[3]

[3]We assume throughout this work the existence of densities and that random variables have finite variance.

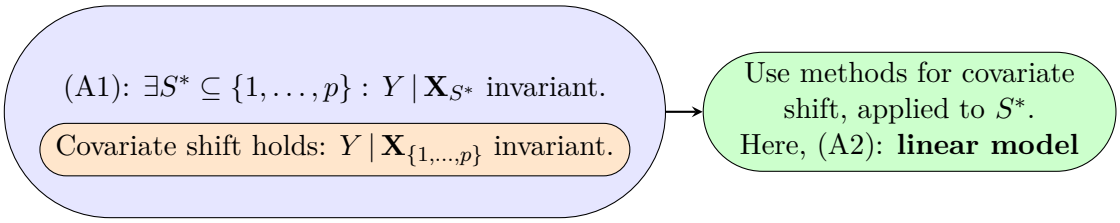


Fig. 3.3 Assumption (A1) (blue) is a relaxation of covariate shift (orange): the covariate shift assumption is a special case of (A1) with $S^* = \{1,\dots,p\}$. Given the invariant set S^*, methods for covariate shift can be applied.

Our main assumption is a relaxation of covariate shift, see Section 3.1.3.1. We propose to learn a feature representation of the tasks which *selects* an invariant subset:

$$h(\mathbf{X}) = \mathbf{X}_{S^*},$$

where S^* leads to invariant conditional mappings $Y^k \,|\, \mathbf{X}^k_{S^*}$ in all tasks. Since no data from the test task are observed in DG, we assume that such invariance also holds in the test task T. The regularity between tasks is made at the level of the *mechanism* $Y^k \,|\, h(\mathbf{X}^k)$ which remains the same in all tasks, and not on the distribution of the features $h(\mathbf{X}^k) = \mathbf{X}^k_{S^*}$ themselves. The distribution of the features $h(\mathbf{X}^k)$ is free to vary arbitrarily between tasks. We now propose a formal statement of our assumptions for DG.

Assumption 11 (Invariant Feature Representations for DG) *We make the following assumptions regarding the tasks $P^1,\dots,P^D,P^T$:*

(A1) There exists a subset $S^* \subseteq \{1,\dots,p\}$ of features such that

$$Y^k \,|\, \mathbf{X}^k_{S^*} \overset{d}{=} Y^{k'} \,|\, \mathbf{X}^{k'}_{S^*} \quad \forall k,k' \in \{1,\dots,D\}. \tag{3.8}$$

*We say that S^* is an **invariant set** which leads to invariant conditionals. Here, $\overset{d}{=}$ denotes equality in distribution.*

(A1') This invariance also holds in the test task T, i.e., (3.8) holds for all $k,k' \in \{1,\dots,D,T\}$.

(A2) The conditional distribution of Y given an invariant set S^* is linear: there exists $\alpha \in \mathbb{R}^{|S^*|}$ and a random variable ϵ such that for all $k \in \{1,\dots,D\}$,

$$[Y^k \,|\, \mathbf{X}^k_{S^*} = x] \overset{d}{=} \alpha^t x + \epsilon^k,$$

that is $Y^k = \alpha^t \mathbf{X}^k_{S^*} + \epsilon^k$, with $\epsilon^k \perp\!\!\!\perp \mathbf{X}^k_{S^*}$ and for all $k \in \{1,\dots,D\}$, $\epsilon^k \overset{d}{=} \epsilon$.

In covariate shift, one usually assumes that (A1') holds for the set of all features. Therefore, (A1') is a weaker condition than covariate shift, see Figure 3.3. We regard this assumption

as a building block that can be combined with any method for covariate shift such as the ones mentioned in Section 3.1.3.1, applied to covariates in the subset S^*. Section 3.1.3.1 also discusses the difficulties encountered when applying covariate shift, especially in terms of estimation. Studying the implications of Assumptions (A1), (A1') and (A2) does not aim to alleviate the difficulties of covariate shift, but rather to elucidate a relaxation of covariate shift which may be more realistic when some of the covariates are causal, and some are not. We concentrate on linear relations (A2), which circumvents the estimation problems discussed in Section 3.1.3.1.

Sections 3.2.3 to 3.2.4 assume that we are given an invariant subset S^* that satisfies (A1) and (A2), and show how the knowledge of S^* can be exploited for DG. Throughout this section, we assume that additionally to (A1) and (A2), assumption (A1') holds. Given that no data from the test task are observed, this is a strong assumption which cannot be tested during training. We believe nonetheless that no statement can be made about DG without making assumptions about the test task and how it relates to the training tasks.

Section 3.2.6 discusses a method for *learning* an invariant subset from data. We further discuss the implications of assumption (A1') when doing estimation in Section 3.2.6.3.

3.2.2 Proposed Estimator

Here and below, we focus on linear regression using the squared loss

$$\mathcal{E}_{P^T}(\beta) = \mathbb{E}_{(\mathbf{X}^T, Y^T)\sim P^T}(Y^T - \beta^t \mathbf{X}^T)^2 \tag{3.9}$$

(the superscript T corresponds to the test task, not to be confused with the transpose, indicated by superscript t). We denote by $\mathcal{E}_{P^1,\dots,P^D}(\beta)$ the squared error averaged over the training tasks $k \in \{1, \dots, D\}$ for coefficient β.

The optimal predictor obtained by minimising (3.9) is the conditional mean

$$\beta^{opt} := \arg\min_{\beta\in\mathbb{R}^p} \mathcal{E}_{P^T}(\beta) = \mathbb{E}\left[Y^T \,|\, \mathbf{X}^T\right], \tag{3.10}$$

which is not available during training.

Given Assumption 11, we know that for all tasks $k \in \{1, \dots, D, T\}$, $Y^k \,|\, h(\mathbf{X})$ has the same distribution, where $h(\mathbf{X}) = \mathbf{X}_{S^*}$ for some invariant subset S^* of features. This implies that the conditional mean given an invariant subset of predictors is a reasonable estimator, which can be approximated by pooling the data in the training tasks and including only features in the invariant subset S^*.

More precisely, let S^* be a invariant subset satisfying Assumption 11. Let $\beta^{S^*} = \arg\min_{\beta\in\mathbb{R}^{|S^*|}}(Y^1 - \beta^t \mathbf{X}^1_{S^*})^2$ be the vector obtained by minimising the squared loss in the training tasks using only predictors in S^*. We propose as a predictor the vector $\beta^{CS(S^*)} \in \mathbb{R}^p$ obtained by adding zeros to β^{S^*} in the dimensions corresponding to covariates outside of

S^*[4]. More formally, we propose to use as a predictor

$$\begin{array}{ccc} \mathbb{R}^p & \to & \mathbb{R} \\ \mathbf{x} & \mapsto & \mathbb{E}[Y^1 \,|\, \mathbf{X}^1_{S^*} = \mathbf{x}_{S^*}] \end{array} \quad \text{and write} \quad \mathbb{E}[Y^1 \,|\, \mathbf{X}^1_{S^*} = \mathbf{x}_{S^*}] = \left(\beta^{CS(S^*)}\right)^t \mathbf{x}. \tag{3.11}$$

Given (A1), the conditional expectation in (3.11) is the same in all training tasks, that is,

$$\forall k \in \{2, \ldots, D\}, \quad \mathbb{E}[Y^k \,|\, \mathbf{X}^k_{S^*} = \mathbf{x}_{S^*}] = \mathbb{E}[Y^1 \,|\, \mathbf{X}^1_{S^*} = \mathbf{x}_{S^*}]. \tag{3.12}$$

In particular, $\beta^{CS} := \beta^{CS(\{1,\ldots,p\})}$ is the estimator obtained when assuming standard covariate shift. The equality of the mappings or *mechanisms* in (3.12) is a fundamental property for the generalisation of our method to difficult scenarios in which (A1') holds true, as it represents the invariant of our method for DG.

Estimation Consider D samples from the source distributions $\mathcal{D}_k = \{(\mathbf{x}^k_i, y^k_i)\}_{i=1}^{n_k}$ for $k \in \{1, \ldots, D\}$. We refer to the pooled training dataset as the sample $\mathcal{D}$ obtained by pooling the samples $\mathcal{D}_1, \ldots, \mathcal{D}_k$ together. Given a subset S^*, $\beta^{CS(S^*)}$ can be estimated by performing linear regression only on features in S^* in the pooled dataset $\mathcal{D}$.

3.2.3 Optimality in an Adversarial Setting

In DG, the test set may be drawn from a distribution which differs arbitrarily from the training tasks. Given the relaxed covariate shift assumption (A1'), the test distribution will be such that $Y^T = \alpha^t X^T_{S^*} + \epsilon^T$, where the noise ϵ^T follows the same distribution as in the training tasks. The joint distribution $P^T(\mathbf{X}^T, Y^T)$ can otherwise vary arbitrarily, which makes DG challenging. In the population case, predictor (3.11) is equal to α, and thus the squared error in the test task equals $\mathbb{E}(\epsilon^2)$, *regardless* of how different the distribution is from the training tasks. In an adversarial setting, predictor (3.11) satisfies the following optimality condition.

Theorem 12 (Adversarial) *Consider* $(\mathbf{X}^1, Y^1) \sim P^1$,..., $(\mathbf{X}^D, Y^D) \sim P^D$ *and an invariant set* S^* *satisfying (A1) and (A2). The proposed estimator satisfies an optimality statement over the set of distributions such that (A1') holds: we have*

$$\beta^{CS(S^*)} \in \underset{\beta \in \mathbb{R}^p}{\arg\min} \sup_{P^T \in \mathcal{P}} \mathcal{E}_{P^T}(\beta),$$

where $\beta^{CS(S^*)}$ *is defined in* (3.11) *and* $\mathcal{P}$ *contains all distributions over* $(\mathbf{X}^T, Y^T)$, $T = D + 1$, *that are absolutely continuous with respect to the same product measure* μ *and satisfy* $Y^T \,|\, \mathbf{X}^T_{S^*} \overset{d}{=} Y^1 \,|\, \mathbf{X}^1_{S^*}$.

[4]The superscript CS is a reference to covariate shift.

We provide a proof for Theorem 12 in a wider, nonlinear setting. First, we introduce the extension of the linear estimator we introduced to a nonlinear setting. Given a subset S^* leading to invariant predictions, the proposed estimator is defined as the conditional expectation

$$f_{S^*} : \begin{array}{ccc} \mathbb{R}^p & \to & \mathbb{R} \\ \mathbf{x} & \mapsto & \mathbb{E}[Y^1 \,|\, \mathbf{X}^1_{S^*} = \mathbf{x}_{S^*}]. \end{array} \tag{3.13}$$

The following theorem states that f_{S^*} is optimal over the set of continuous functions $\mathcal{C}^0$ in an adversarial setting.

Theorem 13 (Adversarial, non-linear) *Consider D tasks $(\mathbf{X}^1, Y^1) \sim P^1, \ldots, (\mathbf{X}^D, Y^D) \sim P^D$ that satisfy Assumption (A1). Then the estimator f_{S^*} in (3.13) satisfies*

$$f_{S^*} \in \operatorname*{arg\,min}_{f \in \mathcal{C}^0} \sup_{P^T \in \mathcal{P}} \mathbb{E}_{(\mathbf{X}^T, Y^T) \sim P^T} \left(Y^T - f(\mathbf{X}^T) \right)^2 ,$$

where $\mathcal{P}$ contains all distributions over $(\mathbf{X}^T, Y^T)$ that are absolutely continuous with respect to the same product measure μ and satisfy $Y^T \,|\, \mathbf{X}^T_{S^*} \stackrel{d}{=} Y^1 \,|\, \mathbf{X}^1_{S^*}$.

Proof Consider a function f that is possibly different from f_{S^*}, see (3.13). For each distribution $Q \in \mathcal{P}$, we will now construct a distribution $P \in \mathcal{P}$ such that

$$\int (y - f(\mathbf{x}))^2 \, dP \geq \int (y - f_{S^*}(\mathbf{x}))^2 \, dQ \,.$$

In this proof, we assume that the probability distributions in $\mathcal{P}$ are absolutely continuous with respect to Lebesgue measure. The extension to the case where they are absolutely continuous with respect to a same product measure μ is straightforward. Let us therefore assume that Q has a density $(\mathbf{x}, y) \mapsto q(\mathbf{x}, y)$. Define P to be the distribution that corresponds to $p(\mathbf{x}, y) := q(\mathbf{x}_{S^*}, y) \cdot q(\mathbf{x}_N)$, where $\mathbf{x}_N$ contains all components of $\mathbf{x}$ that are not in S^*. In the distribution P, the random vector $\mathbf{X}_N$ is independent of $(\mathbf{X}_{S^*}, Y)$. But then

$$\begin{aligned}
&\int (y - f(\mathbf{x}))^2 \, dP \\
&= \int_{\mathbf{x}_N} \int_{\mathbf{x}_{S^*}, y} (y - f(\mathbf{x}_{S^*}, \mathbf{x}))^2 \, p(\mathbf{x}_{S^*}, y) \, d\mathbf{x}_{S^*} \, dy \, p(\mathbf{x}_N) \, d\mathbf{x}_N \\
&\geq \int_{\mathbf{x}_N} \int_{\mathbf{x}_{S^*}, y} (y - f_{S^*}(\mathbf{x}_{S^*}))^2 \, p(\mathbf{x}_{S^*}, y) \, d\mathbf{x}_{S^*} \, dy \, p(\mathbf{x}_N) \, d\mathbf{x}_N \\
&= \int_{\mathbf{x}, y} (y - f_{S^*}(\mathbf{x}_{S^*}))^2 \, q(\mathbf{x}_{S^*}, \mathbf{x}_N, y) \, d\mathbf{x}_{S^*} \, dy \, d\mathbf{x}_N \\
&= \int (y - f_{S^*}(\mathbf{x}))^2 \, dQ.
\end{aligned}$$

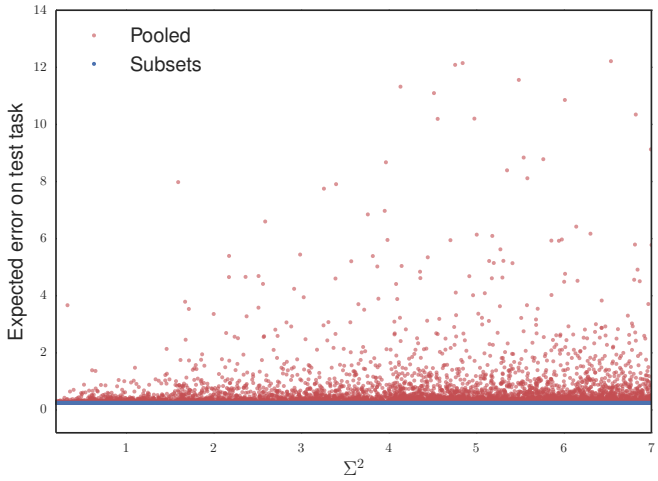


Fig. 3.4 The figure shows expected errors for the pooled approach and the proposed method, see Equation (3.15). $\mu = 0$. We consider two training tasks over $10,000$ simulations. In each, we randomly sample the variance of each covariate in $\mathbf{X}$, the variance of η, and γ. σ^2 is the same in all tasks. As predicted by Proposition Proposition 14 observe that the error from the pooled approach (red) is systematically higher than the error from the prediction using only the invariant subset (blue), and both the error and its variance become large as the variance Σ^2 of coefficients γ^k increases.

■

3.2.4 Comparison Against Pooling the Data

Section 3.2.3 provides an optimality statement for predictor (3.11) when the test set is chosen in an adversarial fashion. In practice, we are often interested in average statements: how well does an estimator perform on average over similar tasks? In our case, how well does predictor (3.11) perform on average over distributions such that (A1') holds true? This question may be especially relevant when we do not expect adversarial test tasks, and rather want to display good performance across the board. We present a result in a specific setting in which predictor (3.11) outperforms all other linear predictors in expectation over tasks.

Let $\mathbf{X}^k_{S*}$ be a vector of independent Gaussian variables in task k. Let the target Y^k satisfy

$$Y^k = \alpha^t \mathbf{X}^k_{S*} + \epsilon^k \,, \tag{3.14}$$

where for each $k \in \{1, \ldots, D\}$, ϵ^k is Gaussian and independent of $\mathbf{X}^k_{S*}$. We have $\mathbf{X}^k = (\mathbf{X}^k_{S*}, Z^k)$, where

$$Z^k = \gamma^k Y^k + \eta^k \,,$$

for some $\gamma^k \in \mathbb{R}$ and where η^k is Gaussian and independent of Y^k. Moreover, assume each training task is equally represented in the training, that is, the training tasks are balanced. We compare properties of estimator $\beta^{CS(S^*)}$ defined in Equation (3.11) against the least squares estimator obtained from pooling the training data, which corresponds to assuming that standard covariate shift holds. Similarly to the toy example in 3.1, the tasks differ in the coefficients γ^k, which are randomly sampled. In particular, the more γ^k varies between tasks, the more these tasks will differ. We prove that the squared loss averaged over unseen test tasks is always larger for the pooled approach, when coefficients γ^k are centred around zero. In the case where they are centred around a non-zero mean, we prove that when the variance between tasks (in this case, for coefficients γ^k) becomes large enough, the invariant approach also outperforms pooling the data.

Proposition 14 (Average performance) *Consider the model described previously. Moreover, assume that the tasks differ as follows: the coefficients $\gamma^1, \ldots, \gamma^D, \gamma^T = \gamma^{D+1}$ are i.i.d. with mean zero and variance $\Sigma^2 > 0$.*[5] *The tasks do not differ elsewhere. In particular, the distribution of $X^k_{S^*}$ is the same for all tasks. Then the least squares predictor obtained from pooling the D training tasks $\beta^{CS} = (\beta^{CS}_{S^*}, \beta^{CS}_Z)$ satisfies:*

$$\mathbb{E}_{\gamma^T}\left(\mathcal{E}_{P^T}\left(\beta^{CS}\right)\right) \geq \mathbb{E}_{\gamma^T}\left(\mathcal{E}_{P^T}\left(\beta^{CS(S^*)}\right)\right) = \sigma^2. \tag{3.15}$$

In particular, this implies the following:

$$\mathbb{E}_{\gamma^1,\ldots,\gamma^D,\gamma^T}\left(\mathcal{E}_{P^T}\left(\beta^{CS}\right)\right) \geq \mathbb{E}_{\gamma^1,\ldots,\gamma^D,\gamma^T}\left(\mathcal{E}_{P^T}\left(\beta^{CS(S^*)}\right)\right) = \sigma^2. \tag{3.16}$$

Moreover, if the coefficients $\gamma^1, \ldots, \gamma^D, \gamma^T$ are i.i.d. with non-zero mean μ, (3.15) *holds for fixed $\gamma^1, \ldots, \gamma^D$ if $\Sigma^2 \geq P(\mu)$, where P is a polynomial in μ.*

Proof We consider three variables and the following generative process: $Y^k = \alpha^t \mathbf{X}^k_{S^*} + \epsilon^k$, $Z^k = \gamma^k Y^k + \eta^k$, where $\epsilon^k \sim \mathcal{N}(0, \sigma^2)$, $\eta^k \sim \mathcal{N}(0, \sigma^2_\eta)$ and $(\mathbf{X}^k_{S^*})_j \sim \mathcal{N}(0, (\sigma_X)^2_j)$. In this model, γ^k is the parameter responsible for the difference between the tasks, while the other parameters are shared between the tasks.

At training time, D tasks are available. We first aim to obtain an explicit formula for the linear regression coefficients $\beta^{CS} = (\beta^{CS}_{S^*}, \beta^{CS}_Z)$ obtained from pooling all the training tasks together. Denote by $\mathbf{X}$, Y and Z the pooled training data. For fixed $\gamma^1, \ldots, \gamma^D$, the

[5] Here, Σ^2 is the variance of the distribution of γ, not to be mistaken with σ^2, the variance of ϵ^k.

expected loss in the training data satisfies for coefficient β verifies:

$$\mathbb{E}\left(\left(Y-(\beta_X)^t\mathbf{X}-\beta_Z Z\right)^2\right)=\frac{1}{D}\sum_{k=1}^{D}\mathbb{E}\left(Y^k-(\beta_X)^t\mathbf{X}^k-\beta_Z Z^k\right)^2$$
$$=\beta_X^t\text{diag}(\sigma_X^2)\beta_X+\frac{\beta_Z^2}{D}\left(\sigma_\eta^2 D+V_Y\overline{\gamma^2}\right)+2(\beta_Z\frac{\bar{\gamma}}{D}-1)\alpha^t\text{diag}(\sigma_X^2)\beta_X+V_Y-2\frac{\bar{\gamma}}{D}V_Y\beta_Z, \tag{3.17}$$

where $V_Y=\alpha^t\text{diag}(\sigma_X^2)\alpha+\epsilon^2$. By differentiating (3.17) with respect to β, we obtain the following expression for the pooled coefficients:

$$\beta_Z^{CS}=\frac{\bar{\gamma}\sigma^2}{V_Y^2\overline{\gamma^2}+D\sigma_\eta^2-\frac{\bar{\gamma}^2}{D}\alpha^t\text{diag}(\sigma_X^2)\alpha}\quad\text{and}\quad\beta_{S^*}^{CS}=(1-\frac{\bar{\gamma}}{D}\beta_Z^{CS})\alpha,$$

where $\overline{\gamma^2}=\sum_{k=1}^{D}(\gamma^k)^2$ and $\bar{\gamma}=\sum_{k=1}^{D}\gamma^k$. Consider now an unseen test task with coefficient γ^T. The expected loss on the test task using the pooled coefficients is:

$$\mathcal{E}_{P^T}(\beta^{CS})=\mathbb{E}\left((Y^T-(\beta_X^{CS})^t\mathbf{X}^T-\beta_Z^{CS}Z^T)^2\right)=\left(\beta_X^{CS}\right)^t\text{diag}(\sigma_X^2)\beta_X^{CS}+(\beta_Z^{CS})^2\left(V_Y(\gamma^T)^2+\sigma_\eta^2\right)$$
$$+2\beta_Z^{CS}\gamma^T\alpha^t\text{diag}(\sigma_X^2)\beta_X^{CS}+V_Y$$
$$-2\alpha^t\text{diag}(\sigma_X^2)\beta_X^{CS}-2\beta_Z^{CS}V_Y\gamma^T.$$

Therefore, the expectation with respect to γ^T is:

$$\mathbb{E}_{\gamma^T}\left(\mathcal{E}_{P^T}(\beta^{CS})\right)=(\beta_X^{CS})^t\text{diag}(\sigma_X^2)\beta_X^{CS}+(\beta_Z^{CS})^2\left(V_Y\Sigma^2+\sigma_\eta^2\right)+V_Y-2\alpha^t\text{diag}(\sigma_X^2)\beta_X^{CS}$$

Denote by $\mathcal{E}_{P^T}(\beta^S)=\sigma^2$ the expected loss when using the invariant conditional predictor $\beta^{S^*}=(\alpha,0)$. Then:

$$\begin{aligned}
&\mathbb{E}_{\gamma^T}\left(\mathcal{E}_{P^T}(\beta^{CS})\right)\geq\mathbb{E}_{\gamma^T}\left(\mathcal{E}_{P^T}(\beta^{S^*})\right)\\
&\Leftrightarrow(\beta_X^{CS})^t\text{diag}(\sigma_X^2)(\beta_X^{CS})+(\beta_Z^{CS})^2\left(V_Y\Sigma^2+\sigma_\eta^2\right)+V_Y-2\alpha^t\text{diag}(\sigma_X^2)\beta_X^{CS}\geq\sigma^2\\
&\Leftrightarrow(\beta_Z^{CS})^2\left(V_Y\Sigma^2+\sigma_\eta^2\right)\geq2\alpha^t\text{diag}(\sigma_X^2)\beta_X^{CS}-(\beta_X^{CS})^t\text{diag}(\sigma_X^2)\beta_X^{CS}-\alpha^t\text{diag}(\sigma_X^2)\alpha\\
&\Leftrightarrow(\beta_Z^{CS})^2\left(V_Y\Sigma^2+\sigma_\eta^2\right)\geq-\frac{\bar{\gamma}^2}{D^2}(\beta_Z^{CS})^2\alpha^t\text{diag}(\sigma_X^2)\alpha,
\end{aligned}$$

by replacing $\beta_X^{CS}=\alpha-\alpha\frac{\bar{\gamma}}{D}\beta_Z^{CS}$. This inequality holds true for any value of the variance Σ^2, and the pooled coefficient leads to larger error in expectation.

Consider now that the coefficients γ^k are fixed and centered around a non-zero value μ. Then the expectation with respect to γ^T of the loss in the test task is the following:

$$\mathbb{E}_{\gamma^T}\left(\mathcal{E}_{P^T}(\beta^{CS})\right) = (\beta_X^{CS})^t \text{diag}(\sigma_X^2)\beta_X^{CS} + (\beta_Z^{CS})^2\left(V_Y(\Sigma^2+\mu^2)+\sigma_\eta^2\right) + 2\beta_Z^{CS}\alpha^t\text{diag}(\sigma_X^2)\beta_X^{CS}\mu + V_Y - 2\alpha^t\text{diag}(\sigma_X^2)\beta_X^{CS} - 2\beta_Z^{CS}V_Y\mu.$$

Then, if $\bar{\gamma} \neq 0$ (if $\bar{\gamma} = 0$, both estimators coincide):

$$\mathbb{E}_{\gamma^T}\left(\mathcal{E}_{P^T}(\beta^{CS})\right) \geq \mathbb{E}_{\gamma^T}\left(\mathcal{E}_{P^T}(\beta^{S^*})\right) \Leftrightarrow \Sigma^2 \geq P(\mu),$$

where $P(\mu) = -\mu^2 - \frac{2}{\beta_Z^{CS}}\left(\left(1-\frac{\bar{\gamma}}{D}\beta_Z^{CS}\right)\frac{\alpha^t\text{diag}(\sigma_X^2)\alpha}{V_Y} - 1\right)\mu - \frac{\bar{\gamma}^2}{V_Y D^2}\alpha^t\text{diag}(\sigma_X^2)\alpha + \frac{\sigma_\eta}{V_Y}$. ■

Figure 3.4 visualises Proposition 14 for two training tasks, it shows the expected errors for the pooled and invariant approaches, see (3.15), as the variance Σ^2 increases. Recall that Σ^2 corresponds to the variance of coefficients γ^k, and thus indicates how different the tasks are. The expected errors are computed using the analytic expression found in the proof of Proposition 14. As predicted by Proposition 14, the expected error of the pooled approach always exceeds the one of the proposed method (the coefficients γ^k are centered around zero), see Equation (3.15). As Σ^2 tends to zero, γ^k is close to zero in all tasks, which explains the equality of both the pooled and invariant errors for the limit case Σ approaching 0. For coefficients γ^k centered around a non zero value, Equation (3.15) does not necessarily hold for small Σ^2.

Proposition 14 presents a setting in which the invariant approach outperforms pooling the data when the test errors are averaged over γ, i.e., $\mathbb{E}_{\gamma^T}\left(\mathcal{E}_{P^T}\left(\beta^{CS}\right)\right) \geq \mathbb{E}_{\gamma^T}\left(\mathcal{E}_{P^T}\left(\beta^{CS}\right)\right)$. It is also clear to see that the equality of the distribution of ϵ^k in Equation (3.14) for all $k \in \{1,\ldots,D\}$ leads to $\text{Var}_\gamma\left(\mathcal{E}_{P^T}\left(\beta^{CS(S^*)}\right)\right) = 0$, thus our invariant estimator minimises the variance of the test errors across all related tasks.

3.2.5 Relation to Causality

Consider the background material and notation introduced in Chapter 2, Section 2.3. Consider a Structural Equation Model (SEM) over variables $(\mathbf{X}, Y)$. Here, we do not specify the graphical relation between Y and the other nodes: Y may or may not have children or parents. Suppose further that the different tasks $P^1,\ldots,P^D$ are intervention distributions of an underlying SEM with graph structure $\mathcal{G}$. If the target variable has not been intervened on, then the set $S^* := \mathbf{PA}_Y^{\mathcal{G}}$ satisfies Assumptions (A1) and (A1'). This means that as long as the interventions will not take place at the target variable, the set S^* of causal parents will satisfy Assumptions (A1) and (A1').

Peters et al. (2016) have given several sufficient conditions for the identifiability of the causal parents in the linear Gaussian framework. As an example, if the interventions take place at informative locations, or if we see sufficiently many different interventions, the set of causal parents is the *only* set S^* that satisfies Assumptions (A1) and (A1'). If there exists more than one set leading to invariant predictions, they consider the intersection of all such subsets. In this sense, seeing more environments helps for identifying the causal structure. In this work, we are interested in prediction rather than causal discovery. Therefore, we try to find a trade-off between models that predict well and invariant models that generalise well to other domains. That is, in the DG setting, we are interested in the subset which leads to invariant conditionals and minimises the prediction error across training tasks.

3.2.6 Learning Invariant Conditionals

We have seen how a known invariant subset $S^* \subseteq \{1, \dots, p\}$ of predictors leading to invariant conditionals $Y^k \,|\, \mathbf{X}^k_{S^*}$, see Assumption 11, can be beneficial in DG, especially when the test task is chosen in an adversarial way. In practice, such a set S^* is not known a priori. We now present a method that aims at *inferring* an invariant subset from data.

We make an explicit distinction between any subset of features, which we denote by S, and an invariant set S^+ (which is not necessarily unique) for which (A1) holds. Such a subset S^+ *does not necessarily satisfy both Assumptions (A1) and (A1').* Indeed, in DG, only (A1) is testable in the training data. More precisely, if several invariant sets which satisfy (A1) are found, we cannot distinguish at training time those which also satisfy (A1'), since no data from the test task are observed. Therefore, given a set of invariant subsets $\{S^+_1, \dots, S^+_\ell\}$, we need to choose a criterion used to select one of these subsets. The method we propose, summarised in Algorithm 1, provides an estimator $\hat{S}$ for an invariant subset S^+, which is chosen as the subset satisfying Assumption (A1) which maximises predictive accuracy on a validation set.

3.2.6.1 Learning an Invariant Subset from Data

Consider D training tasks, a target variable Y^k and a vector $\mathbf{X}^k$ of p features in task k. For $\beta \in \mathbb{R}^p$, we define the residual in task k as:

$$R^k_\beta = Y^k - \beta^t \mathbf{X}^k, \quad k \in \{1, \dots, D\}.$$

By Assumptions (A1) and (A2), there exists a subset S^+ and some vector $\beta^{CS(S^+)}$ such that for all $j \notin S^+$, $\beta^{CS(S^+)}_j = 0$ and $R^1_{\beta^{CS(S^+)}} \overset{d}{=} \dots \overset{d}{=} R^D_{\beta^{CS(S^+)}}$. Such a set S^+ is not necessarily unique. As stated in (Peters et al., 2016), the number of invariant subsets decreases as more different tasks are observed at training time. In Section 3.1.2.3, we introduced a method for testing whether D distributions are equal given D samples. We propose to do an exhaustive

Algorithm 1: Subset search

Inputs: Sample $\{(\mathbf{x}_i^k, y_i^k)\}_{i=1}^{n_k}$ for tasks $k \in \{1, \ldots, D\}$, threshold δ for independence test.

Outputs: Estimated invariant subset $\hat{S}$.

Set $S_{acc} = \{\}$, MSE $= \{\}$.
for $S \subseteq \{1, \ldots, p\}$ **do**
  linearly regress Y on $\mathbf{X}_S$ and compute the residuals $R_{\beta^{CS(S)}}$ on a validation set.
  compute $H = \text{HSIC}_b\left((R_{\beta^{CS(S)},i}, K_i)_{i=1}^n\right)$ and the corresponding p-value p^* (or the p-value from an alternative test, e.g., Levene test.).
  if $p^* > \delta$ **then**
    compute $\widehat{\mathcal{E}}_{P^{1,\ldots,D}}(\beta^{CS(S)})$, the empirical estimate of $\mathcal{E}_{P^{1,\ldots,D}}(\beta^{CS(S)})$ on a validation set.
    S_{acc}.add(S), MSE.add($\widehat{\mathcal{E}}_{P^{1,\ldots,D}}(\beta^{CS(S)})$)
  end
end
Select $\hat{S}$ according to *RULE*, see Section 3.2.6.3 and 3.3.2.1.

search over subsets S of predictors and use this test to establish whether the residuals have the same distribution in all the training tasks.

Among the accepted subsets, we select the subset $\hat{S}$ which leads to the smallest error on a validation set. It is important to note that Algorithm 1 may select features which are non causal, and selects the invariant set which performs well on a validation set, further details are given in Section 3.2.7.2. On the other hand, Peters et al. (2016) care about causal discovery and propose an algorithm with coverage guarantee on the causal parents. This could lead to poor performance in DG: if all tasks are identical, it will choose the empty set as an invariant set and perform mean prediction. In such case, our method outputs the full set of features as an invariant set.

In many prediction problems, the number of features is large and full subset search is not computationally feasible. In Section 3.2.6.2, we propose two solutions for situations where the number of features p is too large for an exhaustive search: a greedy method and variable selection. While the algorithms are presented using linear regression, the extension to a non-linear framework is straightforward. In particular, linear regression can be replaced by a nonlinear regression method.

3.2.6.2 Scalability to a Large Number of Predictors

When the number of features p is large, full subset search is computationally not feasible. We propose two solutions for this scenario. If one has reasons to believe that the signal is sparse, that is the true set S^* is small, one may use a **variable selection** technique such as

Algorithm 2: Greedy subset search

Inputs: Sample $\{(\mathbf{x}_i^k, y_i^k)\}_{i=1}^{n_k}$ for tasks $k \in \{1, \ldots, D\}$, threshold δ for independence test.

Outputs: Estimated invariant set $\hat{S}greedy$.

Set $S_{acc} = \{\}$, $\hat{S}_{current}\{\}$, MSE $= \{\}$.
for $i \in \{1, \ldots, n_{iters}\}$ **do**
  Set $stat_{min} = \infty$.
  for $S \in \mathcal{S}_{\hat{S}_{current}}$ **do**
    linearly regress Y on $\mathbf{X}_S$ and compute the residuals $R_{\beta^{CS(S)}}$ on a validation set.
    compute $H = \mathrm{HSIC}_b\left((R_{\beta^{CS(S)},i}, K_i)_{i=1}^n\right)$ and the corresponding p-value p^* (or the p-value from an alternative test, e.g. Levene test.).
    if $p^* > \delta$ **then**
      compute $\widehat{\mathcal{E}}_{P^{1,\ldots,D}}(\beta^{CS(S)})$, the empirical estimate of $\mathcal{E}_{P^{1,\ldots,D}}(\beta^{CS(S)})$ on a validation set.
      S_{acc}.add(S), MSE.add($\widehat{\mathcal{E}}_{P^{1,\ldots,D}}(\beta^{CS(S)})$),
      set $\hat{S}_{current} = S$.
    end
    else if $H < stat_{min}$ **then**
      set $\hat{S}_{current} = S$, $stat_{min} = H$.
    end
  end
end
Select $\hat{S}greedy$ according to *RULE*, see Sections 3.2.6.3 and 3.3.2.1.

the Lasso (Tibshirani, 1996) as a first step. After a smaller initial set of features has been chosen, Algorithm 1 may be used in the remaining features.

We first point out that the invariant set with best prediction in the training tasks can be assumed to be a subset of the Markov blanket of Y in the corresponding DAG, that is, the set of nodes necessary to predict Y and its children. The intuition is as follows. If the tasks P^k correspond to interventions in an SEM, we may construct an extended SEM with a parent-less environment variable E that points into the intervened variables. Then, P^k equals the distribution of $(\mathbf{X}, Y) \,|\, E = k$, see (Peters et al., 2016, Appendix C). If the distribution of $(\mathbf{X}, Y, E)$ is Markov and faithful with respect to the extended graph, the smallest set S that leads to invariant conditionals and to best prediction is a subset of the Markov blanket of Y: certainly, it contains all parents of Y; if it includes a descendant of Y, this must be a child of Y (which yields better prediction and still blocks any path from Y to E); analogously, any contained ancestor of a child of Y must be a parent of that child.

Thus, if variable screening is satisfied with the Lasso and all relevant variables (and more) are selected, the pre-selection step does not change the result of Algorithm 1 in the limit of infinitely many data. For linear models with L_1 penalisation, variable screening is a well

studied problem, see for instance compatibility and the β_{min} conditions (Bühlmann and van de Geer, 2011, Chapter 2.5).

Alternatively, one may perform a **greedy search** over subsets when full subset search is not feasible. Given a subset S, let $\mathcal{S}_S$ be the collection of sets obtain by adding or removing exactly one of the features in S. $\mathcal{S}_S$ is called the collection of neighbours of S. The algorithm starts by considering the empty set as a candidate. At a given iteration, we proceed as follows:

- if some of the neighbor sets in $\mathcal{S}_S$ are accepted as invariant, we select the one which leads to the smallest validation error,
- if no neighboring set is accepted as invariant, we continue with the subset in $\mathcal{S}_S$ leading to the smallest test statistic (or alternatively, the largest p-value) for the test of equality of distributions.

The resulting algorithm is described in Algorithm 2. As most often for greedy methods, there are no theoretical guarantees.

3.2.6.3 Subset selection in DG

In DG, among the accepted subsets, we select the set $\hat{S}$ which leads to the lowest validation error. Using the notation of Algorithm 1, let S_{acc} be the set of subsets accepted as invariant, and let MSE be the set of their corresponding squared errors on the validation set. The following rule is used for selecting an invariant set in DG.

***RULE* for DG**: Return $\hat{S} = S_{acc}[\arg\min \text{MSE}]$.

Given a set of D training tasks, a collection of sets $\hat{S}_1, \ldots, \hat{S}_u$ (eventually empty) is obtained, all of which lead to accepting the null hypothesis of equality of the distribution of the residuals of the training tasks in DG. Our methods use the MSE on a validation set as a criterion for selecting a subset among these u candidates. This is a design choice which is dependent on the specific application, and can be modified. For instance, if being conservative is important, the MSE may be an inappropriate choice. One may be then interested in combining confidence intervals for the accepted sets. One idea is to consider all accepted sets at the same time, one of which is, with probability $1 - \alpha$, the set S^* from Assumption (A1'). These sets yield different predictions, one of which stems from S^*, again, with probability $1 - \alpha$. In some settings, it might be helpful to output the whole set of predictions. If one is interested in confidence intervals, these may be combined by taking its union. Heinze-Deml et al. (2018) discuss this idea in the context of prediction under interventions.

Remark 15 (Choosing the Size of the Validation Set) *Algorithms 1 and 2 impose a trade-off between fitting a linear model and testing the equality of distributions. Indeed, we*

perform a test for equality of the distribution of the residuals on a validation set, and if more validation data are available the result of the test is more reliable. On the other hand, using too much data for validation leads to a shortage of training data for least squares regression, leading to poor estimation of the regression parameters. This trade-off is common to other problems such as binary two-sample tests (Lopez-Paz and Oquab, 2016).

For linear models, most of the data can be kept for validation, as the parameters can be easily estimated. Careful experimentation on this trade-off would be necessary for non-linear models.

3.2.6.4 Comparison to Other Methods for Transfer

Assumption 11 is motivated by the invariance of causal mechanisms, and focuses on linear settings in which some of the features themselves are causal. The following observations can be made regarding its relation to other common assumptions for transfer, as well as for future directions.

First, many recent methods for transfer learning focus on learning a feature mapping $h : \mathbb{R}^d \to \mathbb{R}^{d'}$ such that the features $h(\mathbf{X}^k)$ contain no information about the task $k \in \{1, \ldots, D\}$. In other words, one cannot predict the task k from which $\mathbf{X}^k$ originates after it has been transformed by h. This assumption performs well in many standard transfer problems. Our assumption is nonetheless orthogonal to this: we aim to learn a feature transformation which simply selects an invariant subset $h(\mathbf{X}) = \mathbf{X}_{S^*}$, and the distribution of $h(\mathbf{X})$ can be arbitrarily different between the tasks. The invariance lies not in the features, but rather in the mechanism mapping input to target. Second, while our work is restricted to linear models, one can extend it by trying to learn non-linear feature representations such that the mapping $Y \mid h(\mathbf{X})$ is the same in all tasks. This would be more realistic in many machine learning settings. For instance, pixels in a vision task are not causal of a target variable, since what is represented by a given pixel can vary arbitrarily between images. Nonetheless, invariance properties could hold with respect to higher level features.

3.2.7 Synthetic Data Experiment

We compare our estimator to different methods, which are summarised in Table 3.2. $\beta^{CS(cau)}$ uses the ground truth for S^* when it is available, $\beta^{CS(\hat{S})}$ corresponds to full search using Algorithm 1, β^{CS} uses the pooled training data and corresponds to assuming standard covariate shift, β^{DICA} performs DICA (Muandet et al., 2013), see Section 3.1.3.3. For DICA, which is a nonlinear method, the kernel matrices are constructed using an rbf kernel, and the length-scale of the kernel is selected according to the median heuristic. β^{mSDA} performs mSDA (Chen et al., 2012) as described in Section 3.1.3.2.

We generate a synthetic data set in which the causal structure of the problem is known. For all experiments, we choose $\delta = 0.05$ as a rejection level for the statistical test in Algorithms 1

estimator	description
$\beta^{CS(cau)}$	Linear regression with true causal predictors (often unknown in practice).
$\beta^{CS(\hat{S})}$	Finding the invariant set $\hat{S}$ using full subset search and performing linear regression using predictors in $\hat{S}$. $\hat{S}greedy$ corresponds to finding the invariant set using a greedy procedure. $\hat{S}Lasso$ corresponds to doing variable selection using Lasso as a first step, then doing full subset search on the selected features.
β^{CS}	Pooling the training data and using linear regression.
β^{dom}	Linear regression using only the available labelled sample from T.
β^{mDA}	Pooling the training data and an unlabelled sample from T, learning features using mSDA (Chen et al., 2012) with one layer and linear output, then using linear regression.

Table 3.2 Methods used in the numerical experiments for DG.

and 2. The sensitivity to the choice of δ is discussed in Section 3.2.7.1. Moreover, we use 40% of the training data to fit the linear models in Algorithms 1 and 2, and the remaining data to test the equality of distributions.

Generative process of the data For each task $k \in \{1, 2, \ldots, D, T\}$, we sample a set of causal variables from a multivariate Gaussian

$$\mathbf{X}_{S*}^k \sim \mathcal{N}(0, \Sigma_{S*}^k)$$

where the covariance matrix Σ_{S*}^k in each task is drawn from a Whishart distribution $\mathcal{W}(U_{S*}^k, p)$, where U_{S*}^k is computed as $V^k(V^k)^t$. Here, V^k is a $|S| \times |S|$ matrix of standard Gaussian random variables.

The target variable Y^k is drawn as

$$Y^k = \alpha^t \mathbf{X}_{S*}^k + \epsilon^k$$

where $\epsilon^k \sim \mathcal{N}(0, 2)$ (the standard deviation of ϵ^k is 6 for the non sparse DG experiment with 30 predictors, see the bottom of Figure 3.5).

We sample the remaining predictor variables as

$$\mathbf{X}_N^k = \gamma^k Y^k + \beta^k (\mathbf{X}_{S*}^k)_C + \eta^k$$

where $\eta^k \sim \mathcal{N}(0, \Sigma_N^k)$. $(\mathbf{X}_{S*}^k)_C$ is a subset of $\mathbf{X}_{S*}^k$ of size $|C|$ which generates both the target Y^k and $\mathbf{X}_N$. γ^k of size $|N|$ is computed as $\gamma^k = (1-\lambda)\gamma_0 + \lambda g^k$, where $\lambda \in [0, 1]$, γ_0 is the same in all tasks while g^k is task dependent. Both γ_0 and g^k are drawn from a standard Gaussian. Similarly to γ^k, β^k is a $(|C|, |N|)$ matrix computed as $\beta^k = (1-\lambda)\beta_0 + \lambda b^k$. Σ_N^k is sampled similarly to Σ_{S*}^k. Finally, α is sampled from a standard Gaussian distribution.

Summary of the results We perform experiments both on a setting where full search is feasible (Figure 3.5, top left) and settings where full search cannot be performed (Figure 3.5,

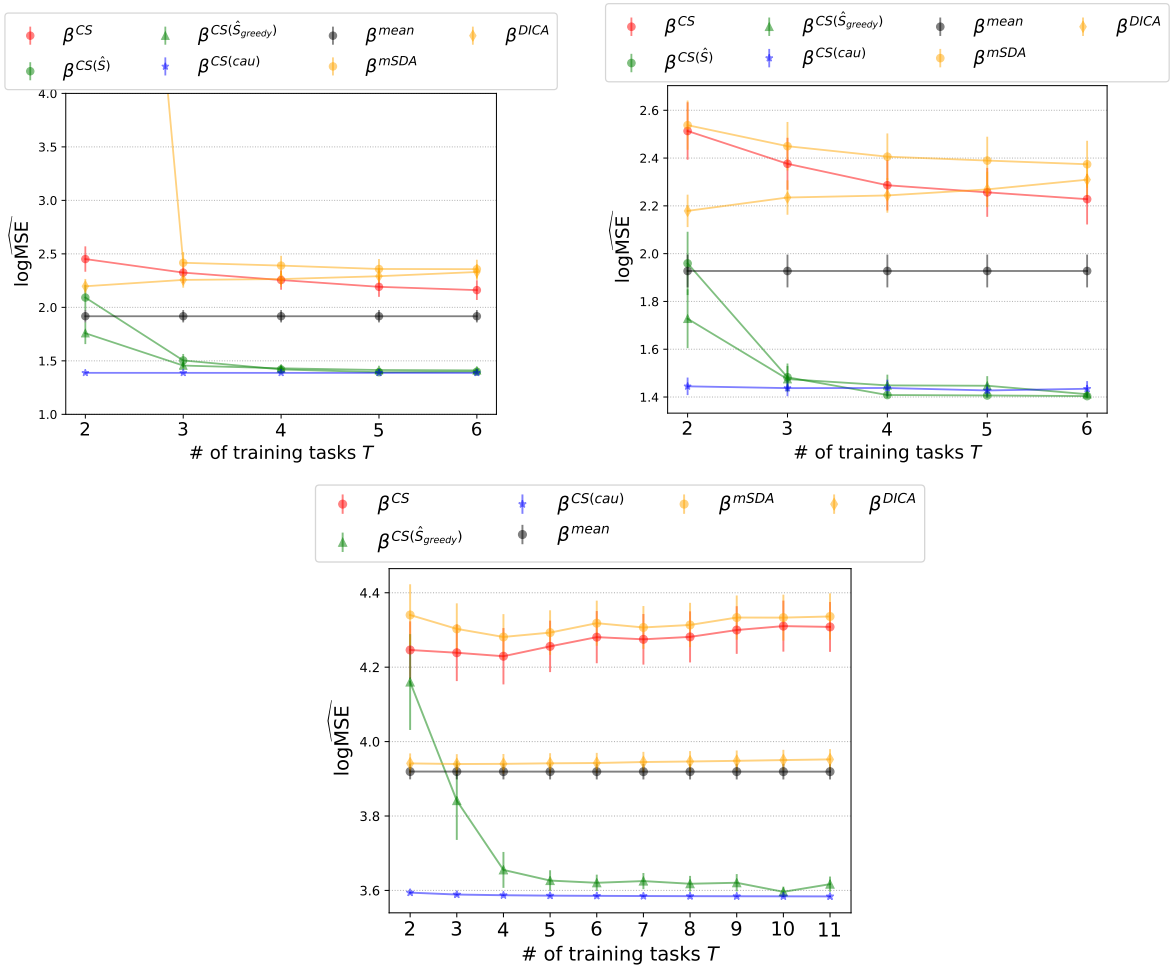


Fig. 3.5 DG setting. Logarithm of the empirical squared error in the test task for the different estimators in the DG setting. The results show averages and 95% confidence intervals for the mean performance over 100 repetitions. We vary the number of tasks D available at training time. *Upper left*: both S and N are of size 3, such that $\mathbf{X}$ is 6-dimensional. $|C|$ is of size one. *Upper right*: 30 noise variables are added to $\mathbf{X}$. Variable selection using the Lasso is used prior to computing $\beta^{CS(\hat{S})}$, while $\beta^{CS(\hat{S}greedy)}$ uses all predictors. *Bottom*: both S and N are of size 15. Full search is not computationally feasible in this setting and only the greedy procedure can be used. Other methods such as β^{CS}, β^{mSDA} and β^{DICA} often perform badly, which explains why in comparison β^{mean} appears to performs well.

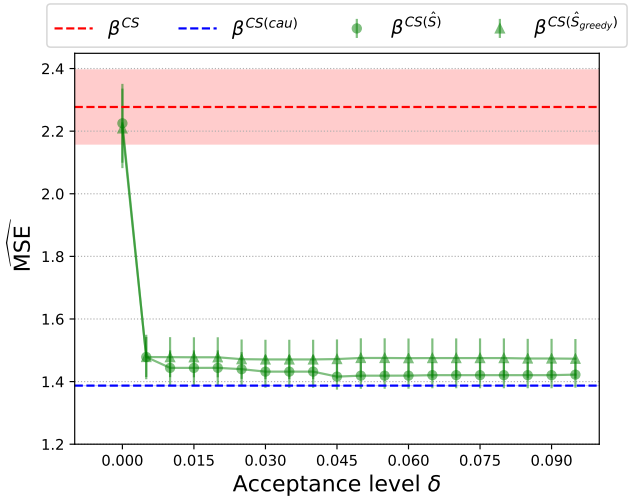


Fig. 3.6 Logarithm of the empirical squared error in the test task in the DG setting as a function of the acceptance level of the statistical test δ in Algorithm 1. The setup corresponds to $T = 3$ in Figure 3.5 (left), also over 100 repetitions. For $\delta = 0$, all subsets are accepted, so the full set of predictors, which minimises the validation squared error, is selected. Algorithm 1 then returns β^{CS}. As δ increases, no subset is accepted, and Algorithm 1 returns the subset with the largest p-value.

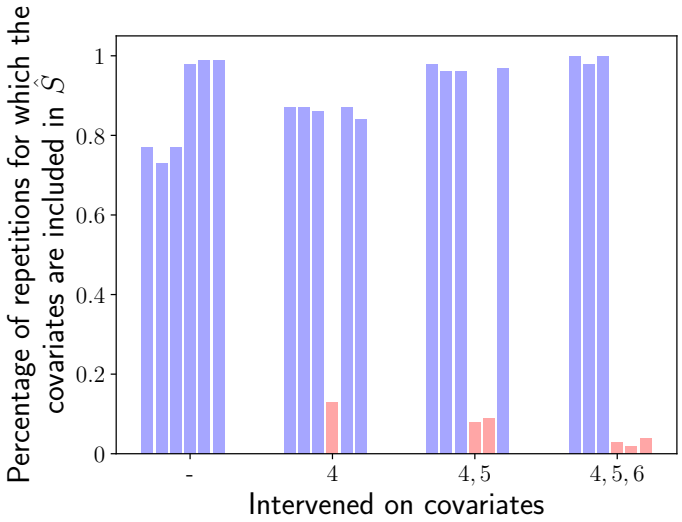


Fig. 3.7 Covariates selected by Algorithm 1 when the training tasks contain interventions only on some of the covariates. The bars represent the percentage of repetitions (out of 100) for which the corresponding covariates were selected. When there are no interventions in the training tasks, meaning that all the training tasks follow the same distribution, Algorithm 1 systematically selects *all* covariates for prediction. When more interventions are performed, however, the corresponding covariates (in red) are excluded in a large number of the repetitions.

top right and bottom). The experiments presented in Figure 3.5 use 4000 examples per training task on the top left and right plots, and only 1000 examples per task on the bottom because of computational reasons. We report the log average empirical MSE over left out test tasks. Two conclusions can be drawn.

i) *How well can we estimate an invariant subset from data?* As the number of training tasks increases, the performance of $\beta^{CS(\hat{S})}$ and $\beta^{CS(\hat{S}greedy)}$ approaches the performance of the ground truth $\beta^{CS(\hat{S})}$. We conclude that invariant subsets can be learned from data and can be better estimated as more training tasks are observed.

ii) *How does invariant prediction perform against other approaches?* When sufficiently many training tasks are observed, methods using an invariant subset outperform the competitors. Moreover, the variance of the error is very small, as the squared error is equal to the variance of ϵ^k which is the same in all the tasks. β^{CS} uses all features for prediction and therefore performs badly, but does get better as the number of tasks increases on the top left and top right. This result suggests the pooled approach still down-weights features when significant variations are observed in the training tasks. Finally, β^{DICA} and β^{mSDA} aim to learn a feature representation of the data in which the training tasks are similar. This invariant does not a priori apply to this problem, on the contrary, the distribution of $\mathbf{X}_{S^*}$ varies quite strongly between tasks, thus possibly explaining the poor performance.

3.2.7.1 Sensitivity to the Acceptance Level δ

Both Algorithm 1 and its greedy version Algorithm 2 receive an acceptance level δ as input for the statistical test. In our other experiments, we chose the standard value of $\delta = 0.05$. Figure 3.6 shows the error on the test tasks for both methods for different values of δ. The setting is the same as in the left of Figure 3.5 for three training tasks. β^{CS} and $\beta^{CS(cau)}$ are provided as reference. For $\delta = 0$, all subsets are accepted as invariant, thus both methods behave like pooling the data. After a critical value of δ, no subset is accepted, and both algorithms return the subset with the largest p-value.

3.2.7.2 Informativeness and Subset Estimation

The estimation of an invariant subset involves finding a subset for which the residuals have the same distribution across tasks. It is desirable, however, that the selected subset is one which explains the data best. This is ensured by selecting the subset which leads to the smallest error on a validation set. Therefore, some features in N may be included in a selected subset *if there are no interventions on this features in the training tasks*, where interventions correspond to changing parameters in the data generating process for the different tasks. More precisely, if including a feature does not lead to a statistically measurable difference in

the distribution of the residuals between the training tasks, it is advantageous in general to include it in the selected subset since the data is better explained.

We illustrate this in Figure 3.7 in a setting with $p = 6$ features. We estimate an invariant subset using Algorithm 1 over 100 repetitions in the following scenarios: i) all the covariates have the same distribution across tasks, ii) one, two or three covariates in N are subject to interventions between the tasks.

We plot the proportion of repetitions for which each of the features is included in the selected subset, see Figure 3.7. We see that, as expected, covariates in N for *which there are no interventions* are included in the selected subset in a large portion of the repetitions, while the other covariates are consistently excluded. This highlights that Algorithm 1 can only exclude covariates whose distribution shifts between training tasks. If being conservative is important for the problem at hand, one can modify Algorithm 1 accordingly, see the end of Section 3.2.6.3.

3.2.7.3 Time Complexity

The most expensive component of Algorithms 1 and 2 is the estimation of the invariant subset. With $n = 4000$ examples available for each of the 6 tasks, and $p = 6$ features, full subset search takes 0.067 seconds and greedy search 0.037, where the results are averaged over 100 repetitions. With $p = 10$, full search averages at 1.57 seconds, and greedy search 0.0396. With $p = 30$, where full search is not feasible, greedy search averages at 1.21 seconds.

3.2.8 Gene Perturbation Experiment

We apply our method to gene perturbation data provided by Kemmeren et al. (2014). This data set consists of the m-RNA expression levels of $p = 6170$ genes $X_1, \ldots, X_p$ of the Saccharomyces cerevisiae (yeast). It contains both $n_{obs} = 160$ observational data points and $n_{int} = 1479$ data points from intervention experiments. In each of these interventions, one known gene (out of p genes) is deleted. In the following, we consider two different tasks. The observational sample is drawn from the first task, and the pooled n_{int} interventions are drawn from the second task.

Motivation. In order to gain an intuition about the experiments we are presenting, consider Figure 3.8. We select as a target a gene Y out of the p genes, and our goal is to predict the activity of Y given the remaining $p - 1$ genes as features. Some of these $p - 1$ genes are causal of the activation of Y. For example, Figure 3.8 shows on the x-axis the activity of two genes (gene A on the left, gene B on the right) such that:

- The expressions of A and B are strongly correlated with the expression of Y.

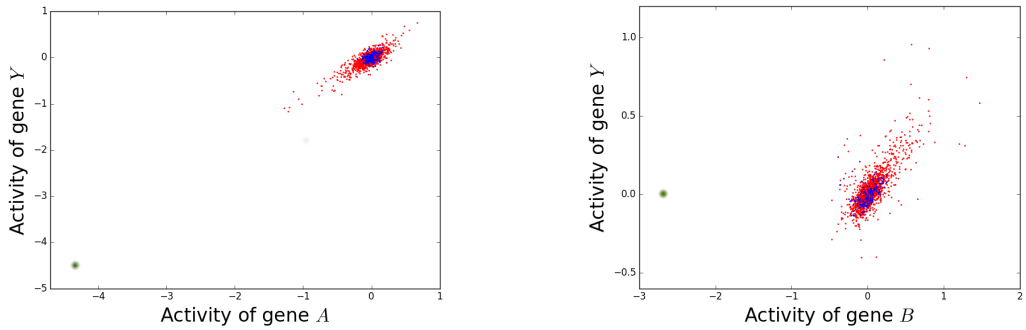


Fig. 3.8 Example of the expression of pairs of genes, where A is causal (left) and B is non-causal (right) of target Y. The blue points are from the observational sample (task 1), the red dots are the interventional sample (task 2), and the green point corresponds to the single interventions in which A and B are intervened on respectively. On the left, a model learned on the data in red and blue would still perform well on the intervention point, which is not the case on the right.

- A is causal of Y (here, we use the definition of a causal effect proposed by Peters et al. (2016)).
- B is non-causal of Y (anticausal or confounded).

In Figure 3.8 (left), the blue points correspond to the 160 data points from the observational sample, which corresponds to the first task. The red dots are the 1478 data points from the interventional sample, except for the single data point for which A is intervened on, and constitute the second task. The plot on Figure 3.8 (right) is constructed analogously for B. We can indeed see that in the pooled sample from task 1 and 2, A and B are both strongly correlated with target Y.

The key difference between both plots are the green points. On Figure 3.8 (left), the green dot corresponds to the single intervention experiment in which gene A is intervened on. Similarly, the green dot on Figure 3.8 (right) is the single point in which B is intervened on. Our goal is to consider the DG setting in which the test task consists on this single intervention point.

For the causal gene A, one expects that a change in the activity of A should translate into a proportional change in the activity of Y. We observe that, in the particular example of the left plot, a linear regression model from A to Y trained only on the pooled data from tasks 1 and 2 (blue and red in Figure 3.8) would lead to a small prediction error on the intervened point (in green). That is, $S^* = \{A\}$ might be a good candidate for a set satisfying Assumptions (A1), (A1') and (A2). For the non-causal gene B, however, intervening on B leaves the activity of Y unchanged, and the linear model learned on the data from tasks 1 and 2 performs badly on the test point in green. In such case, a candidate set is the empty set $S^* = \{\}$, leading to prediction using the mean of the target in the training data. A model

which is aiming to test in these challenging intervention points should therefore include causal genes as features, but exclude non-causal genes. In these experiments, we aim at testing whether we can exclude non-causal genes such as B automatically.

Setup. We address the problem of predicting the activity of a given gene from the remaining genes. We are looking at the following:

- We consider p different **problems**. In each problem $j \in \{1, \dots, p\}$, we aim at predicting the activity $Y = X_j$ of gene j using $(X_\ell)_{\ell \neq j}$ as features.
- In each problem $j \in \{1, \dots, p\}$, two **training tasks** $k \in \{1, 2\}$ are available. The data from the first task is the observational sample, and the data from the second task are all the n_{int} interventions (we shall subsequently remove some points for testing, see below).

The goal is now to apply our method to each of the problems and estimate an invariant subset. Due to the large number of predictors, we first select the 10 top predictor variables using the Lasso and then apply Algorithm 1 to select a set of invariant predictors $\hat{S}$, see $\beta^{\hat{S}Lasso}$ in Table 3.2. We denote the indices of the features selected using Lasso by $L = (L_1, \dots, L_{10})$.

The procedure is then evaluated as follows: for each problem $j \in \{1, \dots, p\}$, we first find the genes in $(X_{L_1}, \dots, X_{L_{10}})$ for which an interventional example is available. Note that this might not hold for all selected genes, since only $n_{int} < p$ interventions are available. We then iterate the following procedure (this is within the context of *the same problem*): for each gene in $(X_{L_1}, \dots, X_{L_{10}})$ for which an intervention is available,

- we put aside the example corresponding to this intervention from the training data (in the motivation example, this would correspond to the green point).
- we estimate an invariant subset $\hat{S} \subseteq L$ using Algorithm 1 with the remaining observational and interventional data.
- we test all methods on the single intervention point which was put aside.

We expect two different scenarios, as explained in the motivation paragraph above: (1) if the intervened gene is a *cause* of the target gene, it should still be a good predictor; then, it should be beneficial to have this gene included in the set of predictors $\hat{S}$. (2) if the intervened gene is anticausal or confounded (we refer to this scenario as *non-causal*), the statistical relation to the target gene might change dramatically after the intervention and therefore, one may not want to base the prediction on this gene. In order to see this effect and understand how the different approaches for DG in Table 3.2 handle the problem, we consider two groups of experiments.

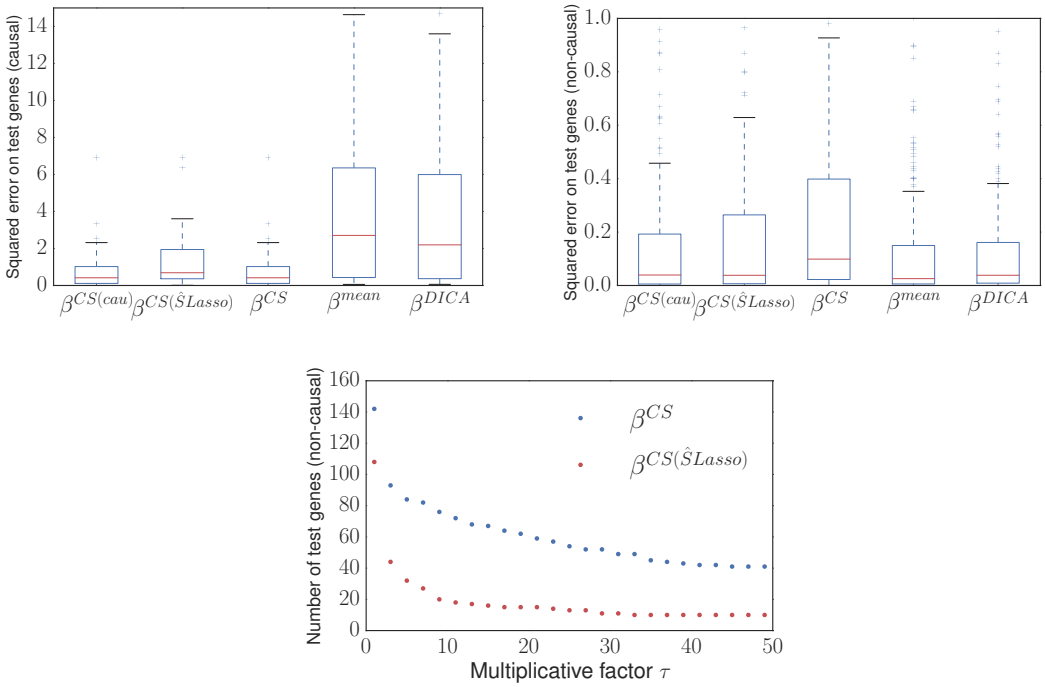


Fig. 3.9 *Top left:* In the causal problems, interventions are performed on causal genes. As expected, the input genes continue to be good predictors, and β^{CS} works well. In the non-causal problems (top right), one of the inputs is intervened upon and becomes a poor predictor, impairing the performance of β^{CS}. The mean predictor β^{mean} uses none of the predictors, and therefore works comparatively well in this scenario. Our proposed estimator $\beta^{CS(\hat{S})}$ provides reasonable estimates in both the causal and non-causal settings, while other methods only perform well in one of the scenarios. β^{DICA} performs similarly to β^{mean} in both scenarios, and is therefore outperformed by other methods in the causal problems (note that β^{DICA} uses all available features). *Bottom*: in the non-causal scenario (2), we plot the number of test genes for which the squared error for β^{CS} is larger than τ times the squared error for $\beta^{CS(\hat{S})}$, and vice-versa, where τ is plotted on the x-axis. This plot shows the number of genes for which one of the method does significantly worse than the other. By this measure, $\beta^{CS(\hat{S}Lasso)}$ outperforms β^{CS} for all values of τ.

(1) we select the target genes Y for which one of the features in L is causal for the activity of Y and for which an intervention experiment is available. 39 problems fall in this causal scenario.

(2) out of the remaining problems we chose target genes with (non-causal) predictors that have been intervened on and — in order to increase the difficulty of the problem — that are strongly correlated with the target gene. We therefore select 269 cases for which a Pearson correlation test (the null hypothesis corresponds to no correlation) outputs a p-value equal to zero.

Results. Figure 3.9 shows box plots for the errors of the different methods for the causal problems (1) on the top left and for the non-causal problems (2) in the top right. We do not plot outliers in order to improve presentation. Figure 3.9 (top left) presents the causal scenario. As expected, pooling does well in this setting. Figure 3.9 (bottom) shows that in the non-causal problems (2), prediction using an invariant subset leads to less severe mistakes on test genes compared to pooling the tasks.

For comparison, since we know which predictors are being intervened on at test time, we included a method that makes use of causal knowledge: $\beta^{CS(cau)}$ uses all 10 predictors in the causal problems (1) and all but the intervened gene in the non-causal problems (2). In practice, this causal knowledge is often not available. We regard it as promising that the fully automated procedure $\beta^{CS(\hat{S}Lasso)}$ performs comparably to $\beta^{CS(cau)}$.

These results reinforce the points made with the synthetic data experiment. From an estimation perspective, Algorithm 1 often manages to recover an invariant subset from the data. In terms of performance, using the knowledge of this invariant subset leads to results robust to whether the intervention is on a causal or non-causal feature, and such model can therefore be deployed with higher confidence in settings where the test task is difficult and distribution shift is severe.

3.3 Learning Representations for Multi-task Learning

We have motivated the use of invariant representations in DG. Indeed we showed that learning an invariant feature mapping $h(\mathbf{X}) = \mathbf{X}_{S^*}$ which selects an invariant subset S^* of features performs well on average over tasks on an example, and performs at its best as the tasks become more different, that is, when distribution shift is stronger. In practice, however, we often observe a small labelled sample $\{(\mathbf{x}_i^T, y_i^T)\}_{i=1}^{n_T}$ from the test task. Multi-task learning (MTL) aims at combining the knowledge obtained from the training tasks with the available sample from the test task to outperform training on the data from the test task alone. As before, we are given an invariant set S^* satisfying (A1) and (A2). In the MTL setting, assumption (A1') is no longer needed as we observe data from the test task and (A1) can be tested from data.

In this section, we aim to build a feature representation $h(\mathbf{X})$ of the original features which outperforms learning on the test data alone. Given Assumption 11, this implies devising a method that combines the test sample with the knowledge of the invariant subset S^* in order to use *all* features for prediction.

Given (A1) and (A2), the target satisfies $Y^k = \alpha^t \mathbf{X}_{S^*}^k + \epsilon^k$, where the noise ϵ^k has zero mean and finite variance, is independent of $\mathbf{X}_{S^*}^k$ and has the same distribution in the different tasks $k \in \{1, \ldots, D, T\}$. Our objective is to use the knowledge gained from the training tasks to get a better estimate of β^{opt} defined in Equation (3.10). We describe below a way to tackle this using missing data methods.

3.3.1 Learning Features Via a Missing Data Approach

In this section, we specify how we tackle MTL by framing it as a missing data problem. In order to motivate the method, assume that for each $k \in \{1, \ldots, D, T\}$, there exists another probability distribution Q^k with density q^k having the following properties: (i) when restricted to $(\mathbf{X}^k_{S^*}, Y^k)$, Q^k coincides with P^k, (ii) the conditional $q^T(y \,|\, \mathbf{x}_{S^*}, \mathbf{x}_N)$ coincides with $p^T(y \,|\, \mathbf{x}_{S^*}, \mathbf{x}_N)$ on the test task and (iii) $q(y \,|\, \mathbf{x}_{S^*}, \mathbf{x}_N) := q^k(y \,|\, \mathbf{x}_{S^*}, \mathbf{x}_N)$ is the same in all tasks. Property (iii) does not hold for distributions P^k, since this would imply that the whole set of features is an invariant set and standard covariate shift holds.

Our goal is to learn a regression model from Y on $\mathbf{X}_{S^*}$ and $\mathbf{X}_N$ in P^T. Given property (ii), this problem coincides with learning the same regression model in Q^T. Moreover, property (iii) implies that we can pool the data from all tasks Q^k since the conditionals match. This is not possible, of course, for the given data, which we have received from the distributions P^k. However, we do have access to data from the marginal distribution $Q^k(\mathbf{X}^k_{S^*}, Y^k)$ for $k \in \{1, \ldots, D, T\}$ given that property (i) holds. If we consider the remaining values $\mathbf{X}^k_N$ as *missing*, and we have a way to estimate them from data, we can exploit properties (i), (ii) and (iii) to learn the conditional mapping $\mathbb{E}\left[Y^T \,|\, \mathbf{X}^T\right]$.

But now assume that in all training tasks, we only have access to the marginal $(\mathbf{X}^k_{S^*}, Y^k)$ from Q^k. Any method that addresses the regression under these constraints be used with the data available because of (i). We first prove the existence of such distributions Q^k:

Proposition 16 (Correctness of transfer) *Let S^* be an invariant set verifying (A1) and (A2). For $k \in \{1, \ldots, D, T\}$, denote by $(\mathbf{x}, y) \mapsto p^k(\mathbf{x}, y)$ the density of P^k. Then there exists a function $q : \mathbb{R}^p \to \mathbb{R}^+$ such that for each $k \in \{1, \ldots, D, T\}$, there exists a distribution Q^k with density q^k such that for all $(\mathbf{x}, y) \in \mathbb{R}^{d+1}$, for all $k \in \{1, \ldots, D, T\}$,*

i) $q^k(\mathbf{x}_{S^*}, y) = p^k(\mathbf{x}_{S^*}, y)$,

ii) $q^T(y \,|\, \mathbf{x}_{S^*}, \mathbf{x}_N) = p^T(y \,|\, \mathbf{x}_{S^*}, \mathbf{x}_N)$,

iii) $q^k(y \,|\, \mathbf{x}_{S^*}, \mathbf{x}_N) = q(y \,|\, \mathbf{x}_{S^*}, \mathbf{x}_N)$.

Proof For $k \in \{1, \ldots, D, T\}$, let Q^k be the probability distribution with density:

$$q^k(\mathbf{x}_{S^*}, \mathbf{x}_N, y) := p^k(\mathbf{x}_{S^*}, y) p^T(\mathbf{x}_N \,|\, \mathbf{x}_{S^*}, y). \tag{3.18}$$

In the test task T, we trivially have $q^T = p^T$. First, it is easy to see that q^k and p^k have the same marginal distribution over $\mathbf{x}_{S^*}$ and y. Indeed:

$$\begin{aligned} q^k(\mathbf{x}_{S^*}, y) &= \int_{\mathbb{R}^{|N|}} q^k(\mathbf{x}_{S^*}, \mathbf{x}_N, y) d\mathbf{x}_N \\ &= \int_{\mathbb{R}^{|N|}} p^k(\mathbf{x}_{S^*}, y) p^T(\mathbf{x}_N \,|\, \mathbf{x}_{S^*}, y) d\mathbf{x}_N \\ &= p^k(\mathbf{x}_{S^*}, y) \int_{\mathbb{R}^{|N|}} p^T(\mathbf{x}_N \,|\, \mathbf{x}_{S^*}, y) d\mathbf{x}_N = p^k(\mathbf{x}_{S^*}, y). \end{aligned} \tag{3.19}$$

Second, we prove that the conditional $q^k(y \,|\, \mathbf{x}_{S^*}, \mathbf{x}_N)$ is the same in all tasks. Indeed, by applying Bayes' rule:

$$\begin{aligned} q^k(y \,|\, \mathbf{x}_{S^*}, \mathbf{x}_N) &= q^k(\mathbf{x}_N \,|\, y, \mathbf{x}_{S^*}) \frac{q^k(y, \mathbf{x}_{S^*})}{q^k(\mathbf{x}_{S^*}, \mathbf{x}_N)} \\ &= p^T(\mathbf{x}_N \,|\, y, \mathbf{x}_{S^*}) \frac{q^k(y \,|\, \mathbf{x}_{S^*})}{q^k(\mathbf{x}_N \,|\, \mathbf{x}_{S^*})} \\ &= p^T(\mathbf{x}_N \,|\, y, \mathbf{x}_{S^*}) \frac{p^k(y \,|\, \mathbf{x}_{S^*})}{\int_{\mathbb{R}} q^k(y, \mathbf{x}_N \,|\, \mathbf{x}_{S^*}) dy} \\ &= p^T(\mathbf{x}_N \,|\, y, \mathbf{x}_{S^*}) \frac{p^k(y \,|\, \mathbf{x}_{S^*})}{\int_{\mathbb{R}} q^k(\mathbf{x}_N \,|\, y, \mathbf{x}_{S^*}) q^k(y \,|\, \mathbf{x}_{S^*}) dy} \\ &= p^T(\mathbf{x}_N \,|\, y, \mathbf{x}_{S^*}) \frac{p^k(y \,|\, \mathbf{x}_{S^*})}{\int_{\mathbb{R}} p^T(\mathbf{x}_N \,|\, y, \mathbf{x}_{S^*}) p^k(y \,|\, \mathbf{x}_{S^*}) dy}. \end{aligned}$$

We have used the fact that $q^k(\mathbf{x}_N \,|\, y, \mathbf{x}_{S^*}) = p^T(\mathbf{x}_N \,|\, y, \mathbf{x}_{S^*})$, which follows from (3.19). Since the last equality leads to a term which is equal in all tasks (indeed, Assumption (A1) ensures that $p^k(y \,|\, \mathbf{x}_{S^*})$ is the same for all $k \in \{1, \ldots, D, T\}$), we have the desired result. ■

Following the previous intuition, for the training tasks $k \in \{1, \ldots, D\}$, we hide the data of $\mathbf{X}_N^k$ and pretend the data in each task $k \in \{1, \ldots, D, T\}$ are drawn from Q^k. Note that some of the data are only missing for the training tasks. More precisely, $\mathbf{X}_N^k$ is missing for $k \in \{1, \ldots, D\}$, while because of (i) in Proposition 16, $(\mathbf{X}_{S^*}^k, Y^k)$ is available for all tasks $k \in \{1, \ldots, D, T\}$. We thus pool the data and learn a regression model of Y versus $(\mathbf{X}_{S^*}, \mathbf{X}_N)$ by maximising the likelihood of the observed data.

We formalise the problem as follows. Let $(\mathbf{z}_i)_{i=1}^n = \{(\mathbf{x}_{S^*,i}, \mathbf{x}_{N,i}, y_i)\}_{i=1}^n$ be a pooled sample of the available data from the training tasks and the test task, in which $\mathbf{x}_{N,i}$ is considered missing if $\mathbf{x}_i$ is drawn from one of the training tasks. Here, $n = \sum_{k=1}^T n_k$ is the total number of training and test examples. Denote by $\mathbf{z}_{obs,i}$ the components of $\mathbf{z}_i$ which are not missing. More precisely, $\mathbf{z}_{obs,i} = \mathbf{z}_i$ if i is drawn from the test task and $\mathbf{z}_{obs,i} = (\mathbf{x}_{S^*,i}, y_i)$ otherwise. Moreover, let Σ be a $(p+1) \times (p+1)$ positive definite matrix, and Σ_i is the

submatrix of Σ which corresponds to the observed features for example i. If example i is drawn from a training task, Σ_i is of size $(|S^*|+1)\times(|S^*|+1)$, and $(p+1)\times(p+1)$ otherwise. The log-likelihood based on the observed data for matrix Σ satisfies:

$$\ell(\Sigma) = \text{const} - \frac{1}{2}\sum_{i=1}^{n} \det(\Sigma_i) - \frac{1}{2}\mathbf{z}_{obs,i}^T \Sigma_i^{-1} \mathbf{z}_{obs,i}, \tag{3.20}$$

and our goal is to find Σ which maximises (3.20). This model for the likelihood assumes that the data is multi-variate Gaussian with covariance matrix Σ.

When all data are observed, the least squares estimator β^{opt} can be seen as the result of a two step procedure. First, (3.20) is maximised for the sample covariance matrix. Then, one computes the conditional mean $\mathbb{E}[Y \mid \mathbf{X} = \mathbf{x}]$ of the estimated joint distribution of $(\mathbf{X},Y)$. In the case of missing data, however, the sample covariance matrix does no longer maximise (3.20), see Section 3.3.1.1 below.

Chapter 11 in Little and Rubin (1986) provides the update equations for optimising Equation (3.20) using Expectation Maximisation (EM) (Dempster et al., 1977). More precisely, given an estimate Σ^r of the covariance matrix at step r, the algorithm goes as follows.

E step: For an example i, we define

$$\mathbf{z}_i^r := \begin{cases} \mathbf{z}_i \text{ if example } i \text{ is from the test task,} \\ (\mathbf{x}_{S^*,i}, \mathbb{E}(\mathbf{x}_N^r \mid \mathbf{z}_{obs,i}), y_i) \text{ otherwise.} \end{cases}$$

Here, we are essentially imputing the data for $\mathbf{x}_N$ in the training tasks by the conditional mean given the observed data, using the current estimate of the covariance matrix Σ^r. The conditional expectation is computed using the current estimate Σ^r and the Gaussian conditioning formula:

$$\mathbb{E}(\mathbf{X}_N^r \mid \mathbf{z}_{obs,i}) = \Sigma_{N z_{obs}}^r (\Sigma_{z_{obs}}^r)^{-1} \mathbf{z}_{obs,i},$$

where $\Sigma_{N Z_{obs}}^r$ is the submatrix of Σ^r corresponding to the cross-covariance between $\mathbf{X}_N$ and $(\mathbf{X}_{S^*}, Y)$, and $\Sigma_{Z_{obs}}^r$ is the submatrix corresponding to the covariance of $(\mathbf{X}_{S^*}, Y)$. For examples from the test task, we simply copy the example, since $P^T = Q^T$. Moreover, define

$$C_{N,i}^r := \begin{cases} 0 \text{ if example } i \text{ is from the test task,} \\ \text{Cov}(\mathbf{x}_N^r \mid \mathbf{z}_{obs,i}) = \Sigma_N^r - \Sigma_{N z_{obs}}^r (\Sigma_{z_{obs}}^r)^{-1} \Sigma_{z_{obs} N}^r \text{ otherwise.} \end{cases}$$

M step: compute the sample covariance given the imputed data:

$$\Sigma^{r+1} = \frac{1}{n}\mathbb{E}\left(\sum_{i=1}^{n} \mathbf{z}_i^r (\mathbf{z}_i^r)^t \mid \mathbf{z}_{obs,i}, \Sigma^r\right) = \frac{1}{n}\sum_{i=1}^{n} \mathbf{z}_i^r (\mathbf{z}_i^r)^t + C_i^r,$$

where C_i^r is a $(p+1) \times (p+1)$ matrix whose submatrix corresponding to features in N is $C_{N,i}^r$, and the remaining elements are 0. The intuition for the M step is simple: we compute the sample covariance with the values imputed for $\mathbf{x}_N$. Since these values are being imputed, matrix C adds uncertainty for the corresponding values.

Once the algorithm has converged, we can read off the regression coefficient from the joint covariance matrix as $\mathbb{E}[Y \mid \mathbf{X}_{S^*} = \mathbf{x}_{S^*}]$. The whole procedure is initialised with the sample covariance matrix computed with the available labelled sample from T.

Incorporating unlabelled data The previous method also allows us to incorporate unlabelled data from the test task. Indeed, assume that an unlabelled sample $\mathbf{x}^T = \{(\mathbf{x}_{S^*,i}^T, \mathbf{x}_{n,i}^T)\}_{i=1}^{n_{ul}}$ from the test task is also available at training time. This can be incorporated in the previous framework since the label Y can be considered to be missing (as opposed to $\mathbf{X}_N^T$ previously). We can then write $\mathbf{z}_i^r = (\mathbf{x}_{S^*,i}, \mathbf{x}_{N,i}, \mathbb{E}(Y_i^r \mid \mathbf{z}_{obs,i}))$ for the unlabelled data, thus imputing the value of y in in the E-step by the conditional mean given $(\mathbf{x}_{S^*,i}, \mathbf{x}_{N,i})$. The added covariance is then $C_{Y,i}^r = \mathrm{Var}(Y)^r - \Sigma_{Y z_{obs}}^r (\Sigma_{z_{obs}}^r)^{-1} \Sigma_{z_{obs} Y}^r$. The rest of the algorithm remains unchanged.

3.3.1.1 A Naive Estimator for Comparison

The following Proposition provides an analytic expression for β^{opt} from (3.10) in terms of α and ϵ.

Proposition 17 *Assume that $\mathbf{X}_{S^*}$ follows an arbitrary distribution and that Assumptions (A1) and (A2) hold. Let $\gamma \in \mathbb{R}^{|N|}$ be the solution of an L^2 regression from $\mathbf{X}_N^T$ on Y^T. Therefore, we can write $\mathbf{X}_N^T = \gamma Y^T + \eta$, with η uncorrelated to Y^T, and the components of η can be correlated. Then the regression coefficients $\beta^{opt} = (\beta_{S^*}^{opt}, \beta_N^{opt})$ minimising the expected squared loss in the test task satisfy*

$$\beta_N^{opt} = \mathbb{E}(\epsilon^2) M^{-1} \gamma \,, \tag{3.21}$$

$$\beta_{S^*}^{opt} = \alpha \left(1 - (\gamma^T)^t \beta_N^T\right) - \Sigma_{X,S^*}^{-1} \Sigma_{X,N} \beta_N \,, \tag{3.22}$$

where $M = \mathbb{E}(\epsilon^2)\gamma\gamma^t + \Sigma_N - \Sigma_{X,N}^t \Sigma_{X,S^*}^{-1} \Sigma_{X,N}$, and $\Sigma_N := \mathbb{E}(\eta\eta^t)$, $\Sigma_{X,S^*} := \mathbb{E}(\mathbf{X}_{S^*}\mathbf{X}_{S^*}^t)$, $\Sigma_{X,N} := \mathbb{E}(\mathbf{X}_{S^*}\eta^t)$ are the corresponding Gram matrices.[6]

[6] We dropped the superscript T to lighten the notation.

Proof To simplify notation, we write Y^T, $\mathbf{X}_{S^*}^T$ and $\mathbf{X}_N^T$ as Y, $\mathbf{X}_{S^*}$ and $\mathbf{X}_N$. We compute the gradients of the expected squared loss after replacing the expression for Y and $\mathbf{X}_{S^*}$:

$$\begin{aligned} L &= \mathbb{E}(Y - \beta_{S^*}^t \mathbf{X}_{S^*} - \beta_N^t \mathbf{X}_N)^2 \\ &= (\alpha(1-\gamma^t\beta_N) - \beta_{S^*})^t \Sigma_{X,S^*} (\alpha(1-\gamma^t\beta_N) - \beta_{S^*}) \\ &\quad + (1-\beta_N^t\gamma)^2 \mathbb{E}(\epsilon^2) + \beta_N^t \Sigma_N \beta_N - 2(\alpha(1-\gamma^t\beta_N) - \beta_{S^*})^t \Sigma_{X,N} \beta_N \end{aligned}$$

The gradients satisfy

$$\begin{aligned} \frac{\partial L}{\partial \beta_{S^*}} &= -2\Sigma_{X,S^*}(\alpha(1-\gamma^t\beta_N) - \beta_{S^*}) + 2\Sigma_{X,N}\beta_N \\ \frac{1}{2}\frac{\partial L}{\partial \beta_N} &= \Sigma_N \beta_N - (1-\gamma^t\beta_N)\mathbb{E}(\epsilon^2)\gamma + \gamma\alpha^t \Sigma_{X,N}\beta_N \\ &\quad - \gamma\alpha^t \Sigma_{X,S^*}(\alpha(1-\gamma^t\beta_N) - \beta_{S^*}) - \Sigma_{X,N}^t(\alpha(1-\gamma^t\beta_N) - \beta_{S^*}) \end{aligned}$$

By setting these to zero, we find the stated values for $\beta_{S^*}^{opt}$ and β_N^{opt}. ■

In the population setting, Proposition 17 provides an expression for β^{opt} as a function of α and ϵ from Assumption (A2). As in the previous paragraph, one could try to estimate the covariance matrix of $(\mathbf{X}, Y)$ using the knowledge of α and ϵ from the training tasks, and then read off the regression coefficients.

In the presence of a finite amount of labelled and unlabelled data from the test task, let $\hat{\Sigma}_{\mathbf{X},Y}$ be the sample covariance matrix in the test task. A naive approach would thus plug in the knowledge of α and ϵ as follows:

- the entries of $\hat{\Sigma}_{\mathbf{X},Y}$ that correspond to the covariances between $\mathbf{X}_{S^*}$ and Y are replaced with $\hat{\Sigma}_{\mathbf{X}_{S^*}} \cdot \alpha$,
- the entry corresponding to the variance of Y is replaced by $\alpha^t \hat{\Sigma}_{\mathbf{X}_{S^*}} \alpha + \mathrm{Var}(\epsilon)$.

This, however, often performs worse than forgetting about α and using the data in the test domain only, see Figure 3.10 (top left). Why is this the case? The naive solution described above leads to a matrix Σ that does not only *not* maximise (3.20) but that often is not even positive definite. One needs to optimise over the free parameters of Σ, which corresponds to the covariance between $\mathbf{X}_N$ and Y, given the constraint of positive definiteness. For comparison, we modified the naive approach as follows. First, we find a positive definite matrix satisfying the desired constraints. In order to do this, we solve a semi-definite Program (SDP) with a trivial objective which always equals zero. Then, we maximise the likelihood (3.20) over the free parameters of Σ with a Nelder-Mead simplex algorithm. The constrained optimisation problem can be shown to be convex in the neighborhood of the optimum (Zwiernik et al., 2017, Sec. 3) if the number of data in the test domain grows.

While gradients can be computed for this problem, gradient-based methods seem to perform poorly in practice (experiments are not shown for gradient based methods).

In an idealised scenario, infinite amount of unlabelled data in the test and labelled data in the training tasks could provide us with $\Sigma_{\mathbf{X}}$, $\Sigma_{(\mathbf{X}_{S^*},Y)}$ and $\mathrm{Var}(Y)$. We could then plug in these values into Σ and optimise over the remaining parameters, see $\beta^{CS(cau+,i.d.)}$ in Figure 3.10 (top left). In practice, we have to estimate $\Sigma_{\mathbf{X}}$, $\Sigma_{(\mathbf{X}_{S^*},Y)}$ and $\mathrm{Var}(Y)$ from data. Thus, the EM approach mentioned above constitutes the more principled approach.

3.3.2 Synthetic Data Experiment

estimator	description
$\beta^{CS(\hat{S}+)}$	Finding the invariant set $\hat{S}$ using full subset search and solve the optimisation problem described in Section 3.3.1.1.
$\beta^{CS(\hat{S}\sharp)}$	Finding the invariant set $\hat{S}$ using full subset search and maximising (3.20) for MTL using EM.
β^{dom}	Linear regression using only the available labelled sample from T.
β^{MTL}	Multi-task feature learning estimator (Argyriou et al., 2007).
β^{CS}	Pooling the training data and using linear regression.
β^{mDA}	Pooling the training data and an unlabelled sample from T, learning features using mSDA (Chen et al., 2012) with one layer and linear output, then using linear regression.

Table 3.3 Methods used in the numerical experiments for MTL.

The generative process for the data in this section is identical to the one used in the experiments in Section 3.2.7. We compare our estimator to different methods, which are summarised in Table 3.3. β^{MTL} performs the Multi-task feature learning algorithm (Argyriou et al., 2007). We combine the subset invariance with task specific information by optimising (3.20) using EM, resulting in regression coefficients $\beta^{CS(\hat{S}\sharp)}$ and $\beta^{CS(cau\sharp)}$ when the ground truth is known. Finally, $\beta^{CS(cau\sharp,UL)}$ indicates that unlabelled data from T was also available. For reference, Figure 3.10 (left) provides results for $\beta^{CS(\hat{S}+)}$ and $\beta^{CS(cau+)}$, which correspond to the estimators obtained by solving the constrained optimisation problem described in Section 3.3.1.1 ($\beta^{CS(cau+)}$ uses the ground truth for S^* and α), while β^{naive} imputes the covariance matrices but does not optimise the free parameters. $\beta^{CS(cau+,i.d.)}$ (infinite data) also assumes that we know the ground truth for the entries of the covariance matrix for the test task corresponding to the covariance of $\mathbf{X}$, the covariance between $\mathbf{X}_{S^*}$ and Y, and the variance of Y.

Summary of the results.

i) *The naive methods from Section 3.3.1.1 require too much data.* In Figure 3.10 (top left), large amounts of labelled data ($36,000$) from the training tasks and unlabelled data from the test task ($50,000$) are available. Both S and N are of size 3, such that $\mathbf{X}$ is 6-dimensional. For all experiments, 6 training tasks are available. We report the

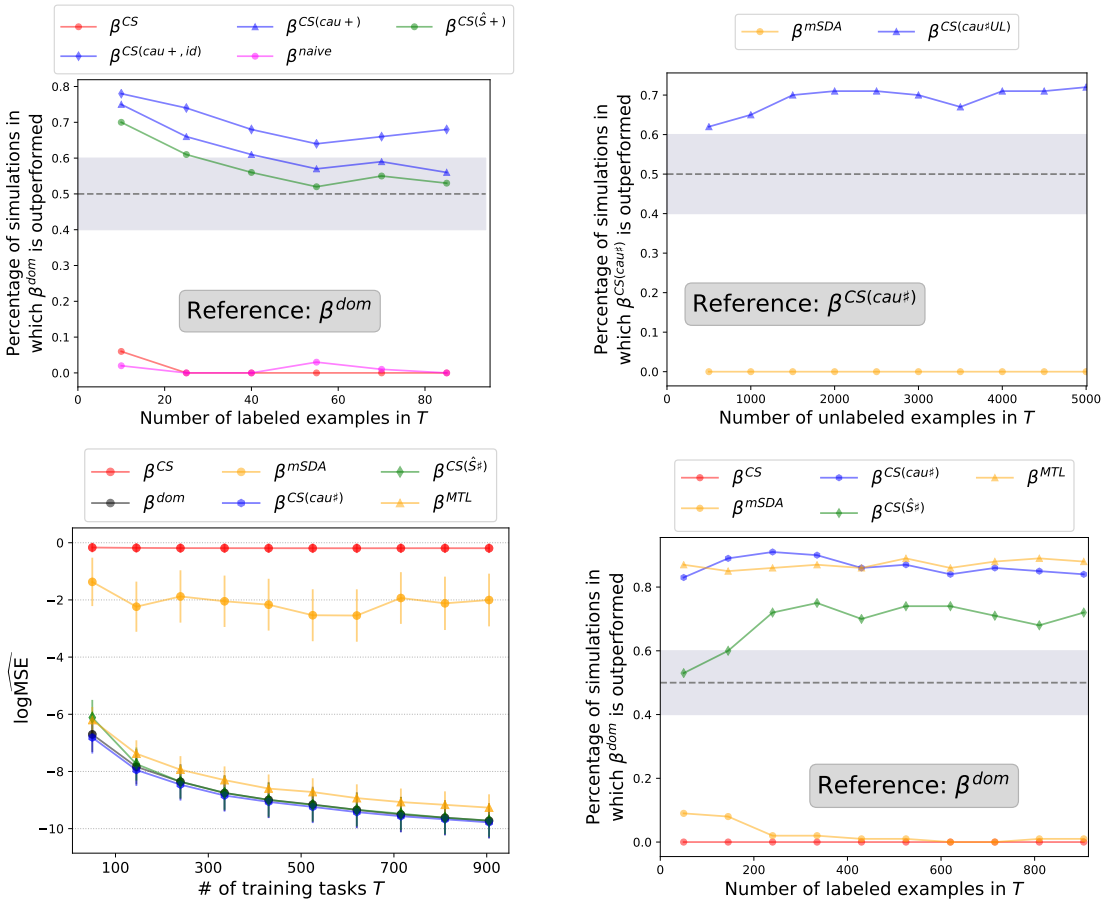


Fig. 3.10 Percentage of repetitions (out of 100) for which the corresponding method outperforms β^{dom} (or $\beta^{CS(cau\sharp)}$ for the top right plot). Both S and N are of size 3, such that $\mathbf{X}$ is 6-dimensional. *Upper left*: This plot shows that the methods $\beta^{CS(\hat{S}+)}$ and $\beta^{CS(cau+)}$ presented in Section 3.3 perform well, but a large amount of data is necessary: $50,000$ unlabelled examples from T and $36,000$ training examples are available. The naive method β^{naive} performs poorly. *Upper right*: we fix the number of training data (500 per task) and vary the amount of unlabelled data available from the test task. We report the percentage of scenarios in which the corresponding method outperforms $\beta^{CS(cau\sharp)}$ this time (which uses no unlabelled data). While β^{mDA} always performs worse than $\beta^{CS(cau\sharp)}$ and does not exploit the unlabelled data, we see that $\beta^{CS(cau\sharp,UL)}$ performs better as the amount of unlabelled data increases. *Bottom*: we vary the number of labelled examples available in each training task. Here, significantly less labelled data was available in the training tasks (from 50 to 1000 per task). In this setting, the methods using unlabelled data were given 100 unlabelled examples. Bottom left: logarithm of the empirical squared error in the test task for different estimators. Bottom right: percentage of repetitions (out of 100) for which the corresponding method outperforms β^{dom}.

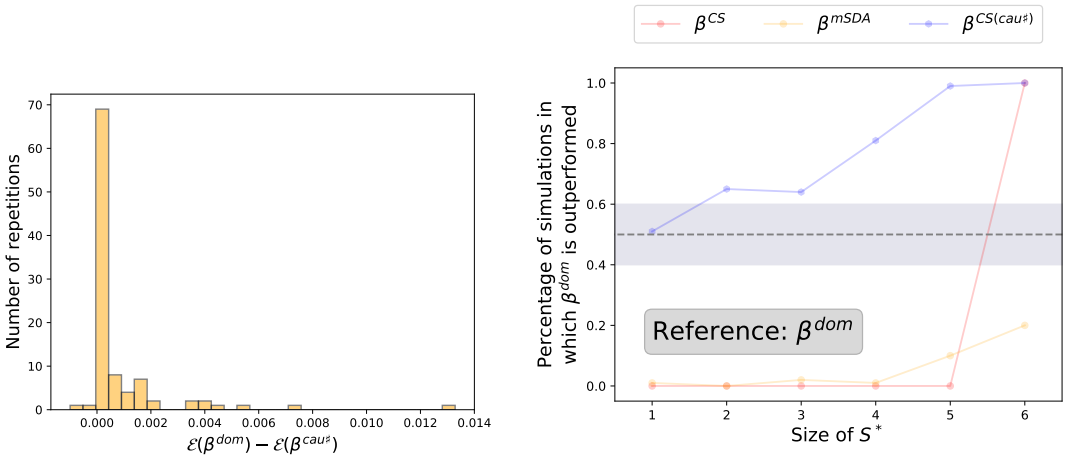


Fig. 3.11 *Left*: 900 examples from each of the training tasks are available (this corresponds to the data point furthest to the right in the bottom plot of Figure 3.10). We run 100 repetitions and plot the histograms of $\Delta = \mathcal{E}(\beta^{dom}) - \mathcal{E}(\beta^{CS(cau\sharp)})$. The proposed estimator outperform β^{dom}: for a large proportion of the repetitions, $\Delta > 0$. More importantly, the distribution of Δ is heavily skewed in the positive values. In other words, when β^{dom} outperforms $\beta^{CS(cau\sharp)}$, the difference in performance is small, while the difference is often larger for the converse. *Right*: setting with 6 tasks and 900 examples per task. We plot the percentage of repetitions (over 100) for which the given methods outperform β^{dom}, as a function of the size of the invariant set S^*. We see that as S^* becomes larger, more information is transferred from the training tasks, and as such the performance of $\beta^{CS(cau\sharp)}$ improves. When S^* is the full set, our method behaves like pooling the data.

percentage of repetitions for which the population MSE of a given approach outperforms β^{dom}. We see that $\beta^{CS(cau+,i.d.)}$ systematically outperforms β^{dom}. Moreover, $\beta^{CS(cau+)}$ and $\beta^{CS(\hat{S}+)}$ also perform well, and positive transfer is effective. However, a prohibitively large amount of labelled and unlabelled data is needed for these approaches, and the differences become non-significant for all methods except $\beta^{CS(cau+,i.d.)}$. This shows the limitation of this family of approaches.

ii) *Our missing data approach is competitive with the best methods when the true invariant subset is known, and still performs well when we learn it from data.* We consider a more realistic setting with only 900 examples per training task, we plot in Figure 3.11 (left) the histogram of the error difference $\Delta = \mathcal{E}(\beta^{dom}) - \mathcal{E}(\beta)$ for $\beta^{CS(cau\sharp)}$. Figure 3.10 (top right) corresponds to the same setting, but we vary the number of unlabelled data available (we only plot methods that use unlabelled data, and $\beta^{CS(cau\sharp)}$ is used as reference instead of β^{dom}). In Figure 3.10 (bottom left) we consider the setting in which only 100 unlabelled data points are available, and only few labelled examples are available in each task. Here, we see that $\beta^{CS(cau\sharp)}$, $\beta^{CS(\hat{S}\sharp)}$ and β^{MTL} perform well, while other methods do not. In terms of MSE (bottom left), the difference in performance between the top competing methods is not statistically significant.

We conclude from these results that when the ground truth for an invariant set is known, our method combines this knowledge with the data from the test task and outperforms using only the data from the task in most repetitions, also performing on par with the state-of-the-art β^{MTL}. Given the uncertainty in correctly estimating an invariant set from data, especially when little data are available in each task, $\beta^{CS(\hat{S}\sharp)}$ performs slightly worse than the ground truth, but still outperforms β^{dom}.

3.3.2.1 Subset Selection in MTL

In DG, among the accepted subsets, we select the set $\hat{S}$ which leads to the lowest validation error on the sample from the test task. In MTL, however, a labelled sample from the test task T is available at training time. Therefore, Algorithm 1 is slightly modified. First, we get all the sets for which H_0 is accepted. Then, we select the accepted set $\hat{S}$ which leads to the smallest 5 fold cross validation error. For each subset, we compute the least squares coefficients using the procedure described in Section 3.3, and measure the prediction error on the held out validation set.

***RULE* for MTL**: Define $CV_{acc} = \{\}$. For each set $S \subseteq S_{acc}$, do CV_{acc}.add(CV_S), where CV_S is the 5-fold cross validation error over the labelled test data obtained by optimising (3.20) using EM with subset S.

Return $\hat{S} = S_{acc}[\arg\min CV_{acc}]$.

3.3.2.2 Informativeness of the Subset Features

We analyse the effect of the informativeness of the subset in the MTL performance. In Figure 3.11 (right) we consider the same setting as in the bottom of Figure 3.10, and we compute the performance against β^{dom} as the size of the invariant set increases. We see that as the size of the invariant set increases, the performance of $\beta^{CS(cau\sharp)}$ improves, since more information is being transferred from the training tasks. When $p = 6$, traditional covariate shift holds, and $\beta^{CS(cau\sharp)}$ performs on par with β^{CS}.

3.3.2.3 Time Complexity

In the MTL experiment in Figure 3.10 (bottom), the EM algorithm runs for 0.00105 seconds on average over 100 repetitions. As a reference, in MTL, linear regression averages at 0.000301 seconds and mSDA at 0.0547 seconds.

3.4 Learning Independent Causal Mechanisms

Sections 3.2 and 3.3 present methods for DG and MTL based on causal assumptions restricted to linear models. In this section, we present a method for transfer learning in a non-linear context inspired by ideas from causality. Let P be a probability distribution on $\mathbb{R}^d$ called

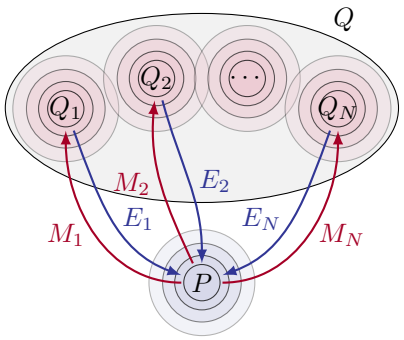


Fig. 3.12 Problem setting. The distributions $Q^1, \ldots, Q^D$ are obtained by applying transformations $M^1, \ldots, M^D$ to a canonical distribution P. Our goal is to learn approximate inverse mappings $E_1, \ldots, E_D$.

the *canonical distribution*. Moreover, we assume that there exist D measurable functions $M^1, \ldots, M^D$ from $\mathbb{R}^d$ to $\mathbb{R}^d$ such that the training tasks are computed as $P^k = M^k(P)$ for $k \in \{1, \ldots, D\}$.[7]

At training time, we observe a sample $\mathcal{D}^P = \{\mathbf{x}_i\}_{i=1}^n \overset{iid}{\sim} P$ and a sample $\mathcal{D} = \{\tilde{\mathbf{x}}_i\}_{i=1}^m \overset{iid}{\sim} Q$, where Q is a mixture distribution of $P^1, \ldots, P^D$. The setup is summarised in Figure 3.12. Our goal is to identify the D functions $M^1, \ldots, M^D$ and approximate their inverse mappings in order to map these example back to their counterpart drawn from P. Given a training task P^k, we wish to learn a feature mapping $h^k : \mathbb{R}^d \to \mathbb{R}^d$ such that if $\mathbf{X} \sim P^k$, $h^k(\mathbf{X})$ is approximately distributed according to P.

Motivating example. Throughout this chapter, we focus on a dataset constructed using MNIST digits. The canonical distribution P is the distribution of standard MNIST digits. The functions M^k correspond to transformations of the digits, such as different translations, contrast inversion and noise addition. At training time, we receive a sample $\mathcal{D}^P$ of standard MNIST digits, and a sample $\mathcal{D}$ of digits transformed by functions M^k. Our goal is to learn a mapping from the transformed digits back to their counterpart from the canonical distribution: a translated digit is to be centred, a noisy digit should be de-noised, etc. This must be accomplished without knowledge of the number or nature of the mechanisms M^k, in a completely unsupervised fashion.

Learning to invert back such mechanisms makes machine learning systems previously trained on P re-usable. Current machine learning paradigms would tend to re-train a system on the transformed digits, instead of leveraging a model already trained to perform well on the canonical MNIST. This is wasteful, as the trained system is not used, and labels are in general expensive to obtain for the new digits. If we know the inverse transformations, we

[7] Each training tasks P^k is defined as the push-forward measure of P induced by M^k.

can simply map the transformed examples back to the canonical distribution, and use the trained model without further modification.

3.4.1 Learning Inverse Feature Mappings with Competition

The training machine is composed of D' parametric functions $E^k : \mathbb{R}^d \to \mathbb{R}^d$ with trainable parameters $\theta_1, \ldots, \theta_{D'}$. We refer to these functions as the *experts*, which should each approximate the feature mapping h^k mapping task k back to the canonical distribution. Note that we do not require $D' = D$, since the number of mechanisms is unknown. The core of our approach relies on a competitive procedure between the experts. Given an input $\widetilde{\mathbf{x}}$, the D' experts produce a candidate version of the canonical counterpart of $\widetilde{\mathbf{x}}$ in P. An objective function $c : \mathbb{R}^d \to \mathbb{R}$ is evaluated on the D' candidates, where c is high in the support of the canonical distribution P, see Section 3.4.1.1 below. Intuitively, $c(E^k(\widetilde{\mathbf{x}}))$ is high if $E^k(\widetilde{\mathbf{x}})$ is likely to be drawn from P.

We compute the value of the objective function on the output of all experts $c^k = c\left(E^k(\widetilde{\mathbf{x}})\right)$, and select the *winning* expert $k^* = \arg\max_k(c^k)$. Based on the objective function c, expert k^* manages the best to map $\widetilde{\mathbf{x}}$ to an example representative of P. The parameters of the winning expert θ_{k^*} are updated as to maximise $c(E^{k^*}(\widetilde{\mathbf{x}}))$, while the parameters of all other experts remain unchanged. This greedy procedure has the goal of encouraging specialisation, as the winning expert gets better at mapping back examples modified by the mechanism which transformed $\widetilde{\mathbf{x}}$.

This greedy procedure aims to optimise a proxy of the following loss function:

$$\theta_1^*, \ldots, \theta_{D'}^* = \arg\max_{\theta_1,\ldots,\theta_{D'}} \mathbb{E}_{\widetilde{\mathbf{X}} \sim Q} \left(\max_{k \in \{1,\ldots,D'\}} c(E^k(\widetilde{\mathbf{X}})) \right). \tag{3.23}$$

3.4.1.1 Using a Discriminator to Compute the Objective Function c

A key component of adversarial training for generative models is a *discriminator* function. In standard generative adversarial networks (Goodfellow et al., 2014), or GANs, the discriminator $D : \mathbb{R}^d \to [0, 1]$ receives as input an image and should output a high score for images drawn from the true data distribution P_D and a low score for data generated by a generator function G, trained to "fool" the discriminator.

We use as an objective function $c : \mathbb{R}^d \to \mathbb{R}$ the score of a discriminator network. This discriminator is trained to distinguish examples drawn from the true canonical distribution P from examples generated by the experts. In this context, the experts may be seen as GANs conditioned on an input example rather than a noise vector. The function c thus depends on parameters θ_c which are jointly trained with the experts: the discriminator is trained to distinguish the output of all the experts from canonical examples, and only the winning expert k^* is updated using the gradients provided by D. The discriminator is trained to

Algorithm 3: Learning independent mechanisms using competition of experts and adversarial training

Inputs: $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^{n} \overset{iid}{\sim} P$; $\widetilde{\mathcal{D}} = \{\widetilde{\mathbf{x}}_i\}_{i=1}^{m} \overset{iid}{\sim} Q$; discriminator network D_{θ_c}; N': number of experts; T: maximum number of iterations;

initialise experts as approximately identity:

$\{E_i \leftarrow \text{TrainAsIdentityOn}(\widetilde{\mathcal{D}})\}_{j=1}^{D'}$

for $t \leftarrow 1$ *to* T **do**

  Sample minibatches:

  $x, \widetilde{x} \leftarrow \text{Sample}(\mathcal{D}), \text{Sample}(\widetilde{\mathcal{D}})$

  Scores from D for all outputs from the experts:

  $\{c_j \leftarrow D(E^k(\widetilde{x}))\}_{j=1}^{D'}$

  Update D: $\theta_D^{t+1} \leftarrow \text{Adam}\Big(\theta_D^t, \nabla \log D(x) + \nabla(1/N' \sum_{j=1}^{N'} \log(1 - c_j))\Big)$

  Update experts: $\{\theta_{E_j}^{t+1} \leftarrow \text{Adam}(\theta_{E_j}^t, \nabla \max_{j \in 1,\ldots,N'} \log(c_j))\}_{j=1}^{N'}$

end

Fig. 3.13 The top row contains 16 random inputs to the networks drawn from the distribution of shifted digits Q. The bottom row shows the outputs from the highest scoring expert for the corresponding input. The system has been trained for 1000 iterations. The digits in the second row look like standard MNIST digits. Moreover, the digit is indistinguishable from the corresponding digit on the first line.

optimise the following loss function:

$$\max_{\theta_c} \Big(\mathbb{E}_{\mathbf{X} \sim P} \log(D_{\theta_c}(\mathbf{X})) + \frac{1}{D'} \sum_{j=1}^{D'} \mathbb{E}_{\widetilde{\mathbf{X}} \sim Q} \big(\log(1 - D_{\theta_c}(E^j(\widetilde{\mathbf{X}}))) \big) \Big), \tag{3.24}$$

where the first term encourages a high score for true canonical examples drawn from P and the second encourages a small score for examples generated by the experts. The algorithm is summarised in Algorithm 3.

3.4.2 Experimental Results

We perform experiments where the canonical distribution P is the distribution of MNIST handwritten digits, and the distribution Q is composed of 10 distribution $P^1, \ldots, P^{10}$ corresponding to 4 pixel translations in eight directions (four diagonals, up, down, left and right) as well as contrast inversion and addition of noise. In order to ensure no matching ground

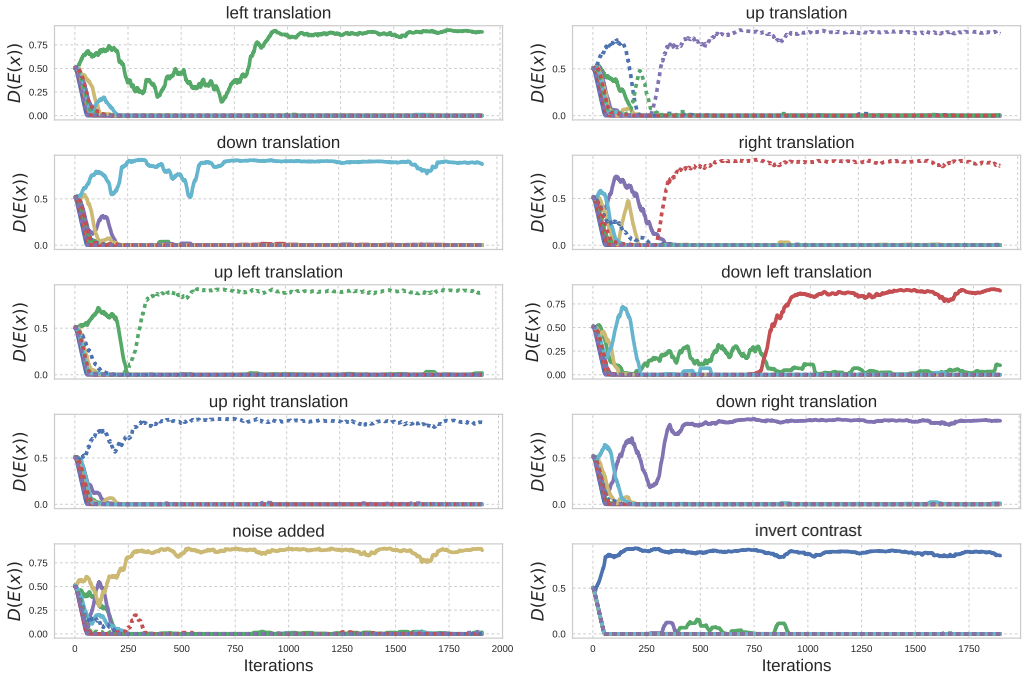


Fig. 3.14 Value of the discriminator score of the experts on data from all transformations. Each line color/style represents one expert. For each of ten different mechanisms (top left to bottom right), the experts are being fed transformed digits. Each expert learns to specialise on a different mechanism, as shown by the score approaching 1. Each curve is smoothed with a moving average of 50 iterations.

truth between examples in P and Q, we split MNIST in half and only transform those in the first half, while the second half is used as canonical digits.

As a preprocessing, we scale the pixel intensities to lie in $[0, 1]$ and zero-pad the digits into a 32×32 image prior to modifying the canonical digits by the respective mechanism.

A key element for the specialisation of the experts is their initialisation. As a first step, we initialise the weights of the experts randomly, and then train them to approximate the identity in Q as an additional pre-training step. In other words, given an input $\mathcal{D} \sim Q$ randomly drawn from the transformed digits, the experts are pre-trained to minimise the squared reconstruction loss $\mathcal{L}_r = \sum_{i=1}^{m}(E^k(\widetilde{\mathbf{x}}_i) - \widetilde{\mathbf{x}}_i)^2$. We found that this improved the speed and stability of training for reason we enumerate below.

Each iteration of training comprises the following. A mini-batch of size 32 drawn from the transformed digits Q is fed to all the experts in parallel. Each expert produces a candidate canonical digit, and the discriminator outputs a score for each candidate. Only the expert with the highest scoring candidate gets updated, and the discriminator additionally receives a mini-batch of canonical digits to update its parameters. Each experiment is run over 2000 such iterations. The experiments are repeated 10 times with different random initialisations for the weights. Details on the architecture of the experts and the discriminator are provided

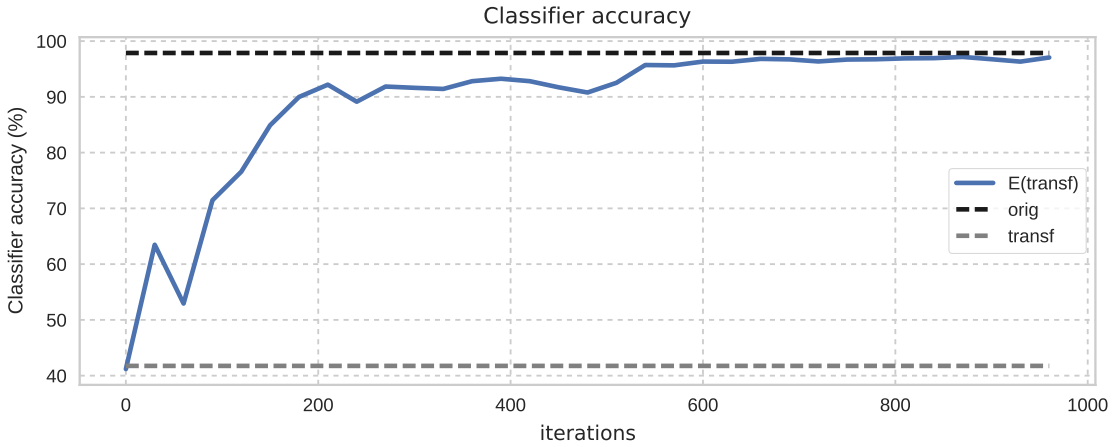


Fig. 3.15 Accuracy of a pretrained CNN MNIST classifier on transformed test digits $\mathcal{D}$ after being mapped back from our method, as a function of the number of training iterations. Our system manages to invert the transformations, with the classifier accuracy quickly approaching the accuracy for digits from the canonical distribution. Note that 600 iterations correspond to having seen about a third of the dataset. The dashed line indicates the test accuracy of the sample $\mathcal{D}^P$ from the canonical distribution of MNIST digits.

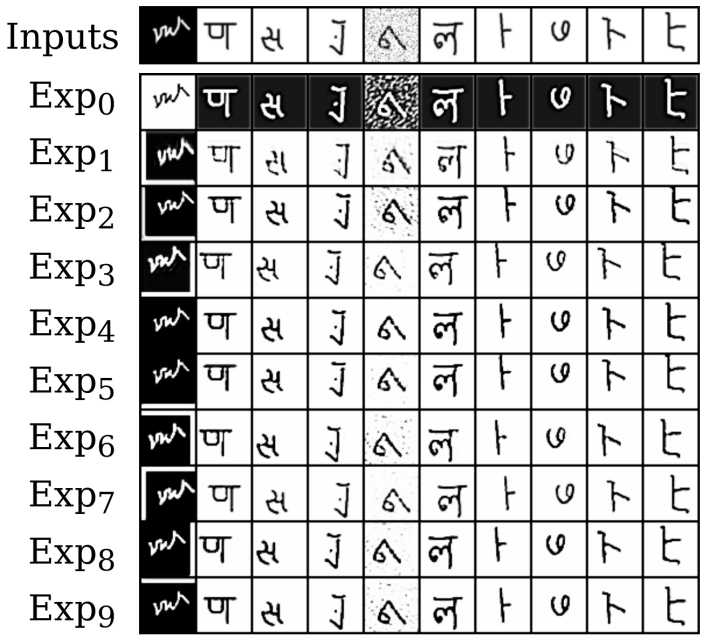


Fig. 3.16 Each column shows how each expert transforms the input presented on top. We arrange the tasks such that the diagonal contains the highest scoring expert for the input given at the top of the column. The experts have learned the inverse mechanisms, consistently applying them to previously unseen symbols.

in Appendix A. We assume that $D' = D$, i.e., that the number of experts is the same as the true number of mechanisms. The setting in which these differ is discussed in Parascandolo et al. (2018). The experts correctly specialised on inverting exactly one mechanism each in 7 out of the 10 runs; in the remaining 3 runs the results were only slightly suboptimal: one expert specialised on two tasks, one expert did not specialise on any, and the remaining experts still specialised on one task each, thus still covering all the existing tasks. Figure 3.13 shows a randomly selected batch of inputs and corresponding outputs from the model. Each independent mechanism was inverted by a different expert. We draw three main conclusions from these experiments.

Each expert inverts a unique mechanism. After an initial phase of competition between the experts, each expert converges to a high score for the discriminator D on exactly one mechanism, see Figure 3.14. Each expert thus specialises in mapping digits modified by one of the transformations exclusively back to canonical digits. We plot the scores assigned by the discriminator for each expert on one of the random runs. Each expert is represented with the same color and linestyle across all 10 tasks. Most experts specialise quickly around 100 iterations. Others, such as the expert depicted in thick green, try to learn two similar tasks (down and down-left translations) simultaneously until iteration 750, but eventually specialise.

A standard MNIST classifier can be used on the transformed digits. We consider a classifier trained on standard MNIST and achieving around 99% test accuracy. As training progresses, we feed-forward all test transformed digits through the experts, select the winning expert and feed the corresponding generated digit to the MNIST classifier. The accuracy of this classifier on the transformed digits increases during training to reach close to 99% accuracy, see Figure 3.15, corresponding to an upper bound on achievable performance. At the beginning of training, 40% accuracy is achieved since the digits are initialised as the identity, and thus do contain information about the classes despite a strong distribution shift.

The experts learn mechanisms that generalise to other distributions. The inverse transformations learned by the experts generalise to other datasets for which no data are observed during training. Following training on MNIST as described above, we transform symbols from Omniglot (Lake et al., 2015) rescaled to 46×46 pixels using the same 10 mechanisms. The experts manage to transform these symbols back to their canonical Omniglot counterpart, see Figure 3.16. A priori, the experts have no reason to generalise. One possible strategy for the experts is to map a transformed digit back to a canonical digit, regardless of which digit this is. In other words, an input 9 with noise addition could be transformed into a canonical 1. The expert could also predict the class of the digit, and output a canonical digit of the same class, which is not the same as in the input. Both these things do not happen,

Fig. 3.17 First row: Omniglot letters that were transformed by a combination of transformations (noise, contrast inversion and translation up left). Second to fourth row: application of denoising, contrast inverting and right down translating experts. Last row: ground truth. Although the experts were not trained on a combination of mechanisms nor on Omniglot letters, they can be used to recover the original symbols.

and the experts even perform well out of sample. We attribute this success to several design choices of the procedure. First, the experts are initialised as the identity, which encourages not deviating strongly from the example. Possibly the easiest way to output a canonical digit is simply to invert the transformation. Second, the experts are fully convolutional and are of moderate size, meaning that complex arbitrary mappings are unlikely to be learned.

Further experiments provide insight about the generalisation of the learned mechanisms. First, we show that the learned inverse mechanisms can be combined. A main drawback of the current training procedure is that the digits are modified by only one transformation. Ideally, a sequence of transformations could be applied, but the nature of the greedy procedure renders this difficult during training. We test whether the trained experts could undo several transformations applied at once. For simplicity, we assume we know which transformations were used. In Figure 3.17, we test on Omniglot letters transformed with three consecutive transformations (noise, up left translation, contrast inversion) by applying the corresponding experts previously trained on MNIST, and correctly recover the original letters.

Second, we analyse the effect of initialising the experts randomly instead of pre-training them to the identity. With random initialisation of the experts, several experts fail to specialise. Out of 10 new runs with random initialisation, only one experiment had arguably good results, with eight experts specializing on one task each, one on two tasks, and the last on none. The performance was worse in the remaining runs. The problem was not that the algorithm takes longer to converge following a random initialisation, as with an additional experiment for $10,000$ iterations the results did not improve. Instead, the random initialisation can lead to one expert winning examples from many tasks at the beginning of training, in which case it is difficult for the others experts to compete for data points.

Third, we notice that training a single network instead of a committee of experts makes the problem more difficult to solve. Using identical training settings, we trained a single network with 32, 64, and 128 filters per layer respectively. None of the runs learned the inverse mechanisms.[8] Note that a single network with 128 filters per layer has about twice as many parameters overall as the committee of 10 experts with 32 filters per layer each. We also tried i) random initialisation instead of the approximate identity, ii) reducing the learning rate of the discriminator by a factor of 10, and iii) increasing the receptive field by adding two pooling and two up-sampling layers, without any improvement. While we do not exclude that careful hyper-parameter tuning may enable a single network to learn multiple mechanisms, we believe a competitive procedure between experts is more appropriate to solve this problem.

Finally, requiring $30,000$ examples from the canonical is too large for many applications. We reduce the number of examples from the original distribution to 64, and find that all experts still specialise and recover good approximations of the inverse mechanisms, using the exact same training protocol. Although the output digits turn out less clean and sharp, we still achieve 96% accuracy on the pre-trained MNIST classifier.

Limitations and further directions The current experimental setup leaves some questions unanswered for which more research is needed. First, the nature of the transformations and the choice of model class for the experts leads to some inductive bias. Indeed, translation, noise additions and contrast inversion are such that a fully convolutional network can invert them without much difficulty. We did not include simple transformations such as rotations since they cannot be inverted within this model class. Adding fully connected layers to the experts increases their capacity and may lead to generalisation problems.

Second, the supports of the transformed digits are reasonably disjoint, which may lead to an easier specialisation. Experiments on different data modalities, such as audio, or on image transformations for which the supports strongly overlap, may lead to increased difficulty in specialisation. Nonetheless, we find encouraging that the experts specialise in this initial experiment, and particularly that they generalise to very different, unseen distributions.

3.5 Conclusion

This chapter revisited the problem of distribution shift by relating the different tasks via causal assumptions. For domain generalisation, where no data from the test task are observed during training, we showed that learning an invariant representation of the data leads to optimality in an adversarial setting, and provided experimental support in simulated and real data. The learned representations for domain generalisation lead to an invariant mapping, yet the distribution of the features themselves can differ between tasks. This partially

[8] Specifically, the network performs well on the contrast inversion task, and poorly on all others.

explains the failure of other methods in our experiments and motivates further research in this direction: representations transfer via the mechanism mapping them to the target, and not the distribution of the representations themselves. While these assumptions are intended for domain generalisation, we provide an algorithm for combining the knowledge of an invariant representation with data from the test task in multi-task learning. Experiments on synthetic data show that especially on the small data regime, the combined representations transfer well and outperform learning only on the test task.

Finally, we provide an algorithm for mapping examples modified by different mechanisms back to their counterpart drawn from a canonical distribution under which a classifier has been trained. We show that the learned inverse mappings generalise to completely different datasets and can thus be re-used as independent modules in other classification tasks.

Chapter 4

Learning Representations for Few-shot Learning

4.1 Introduction

This chapter focuses on learning representations which transfer information from a large dataset to a dataset where few examples are available for each class. Performance in supervised learning problems has increased significantly with the proliferation of deep learning methods, with the downside that significant amounts of data are required for training. The question on how to leverage deep learning systems when learning with very few examples per class remains a highly debated research question.

Humans learn to recognise new objects from only a handful of labelled examples. This success may be due to several reasons, the first being that the powerful visual features produced by our brains are very efficient at characterising visual scenes and allow us to easily discriminate between objects. However, it is also likely that information about previously learned objects is also leveraged when learning new concepts: when discovering a new type or aircraft, leveraging the notions of wings and rotors is likely to facilitate learning. Similarly, we may need to observe significantly more instances of a completely unknown object, abundant in features which are foreign to us.

The goal of this chapter is to exploit both these ideas to build machine learning systems for few-shot learning. More precisely, in situations in which the new classes share similarities with the classes available in an initial training phase, we propose to use the initial classes to build a strong feature extractor using standard deep learning techniques. We then leverage these feature representations of the inputs of the new classes and information on the initial classes for few-shot learning.

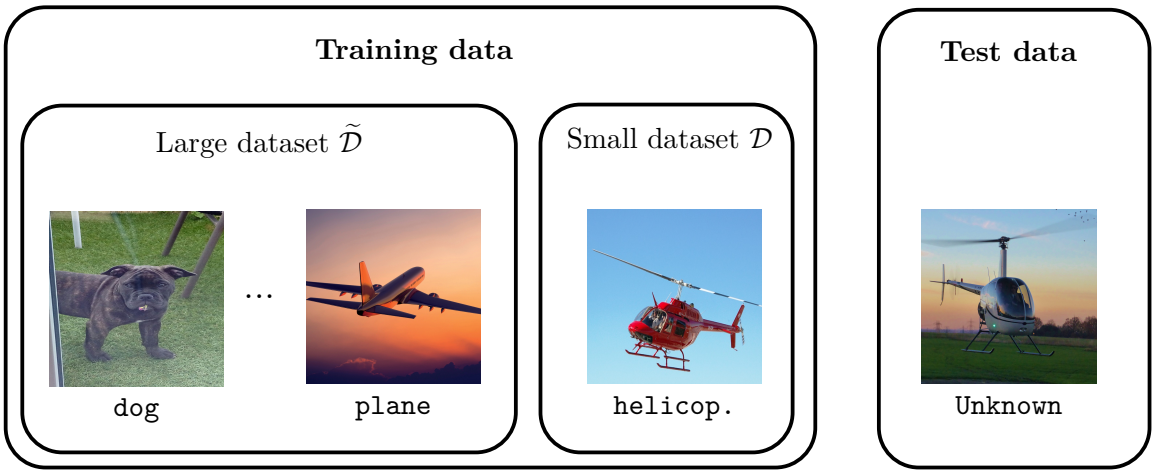


Fig. 4.1 Summary of the few-shot learning task. At training time, a large dataset $\widetilde{\mathcal{D}}$ of labelled examples is available, as well as a smaller dataset $\mathcal{D}$ with only few examples per class. The goal of few-shot learning is to predict on examples drawn from classes found in the small dataset $\mathcal{D}$. Here, we aim to perform well at predicting helicopters, having previously seen only a handful of examples.

4.1.1 Few-shot Learning Setting

Few-shot learning happens in two steps: first, a sample $\widetilde{\mathcal{D}} = \{(\widetilde{\mathbf{u}}_i, \widetilde{y}_i)\}_{i=1}^{\widetilde{N}} \overset{iid}{\sim} P(\mathbf{U}, \widetilde{Y})$ from the training task is available, where the features $\mathbf{U}$ are in $\mathbb{R}^d$ and the label $\widetilde{Y} \in \{1, \ldots, \widetilde{C}\}$. In image classification, $\mathbf{U}$ is a vector of pixel-intensities and $\widetilde{C}$ distinct classes are available for training. Second, we receive a sample from the test task $\mathcal{D} = \{(\mathbf{u}_i, y_i)\}_{i=1}^{N} \overset{iid}{\sim} P^T(\mathbf{U}, Y)$, where the classes $Y \in \{\widetilde{C}+1, \widetilde{C}+C\}$ are different from the training classes. To avoid confusion, we refer to the test task P^T as the few-shot distribution. Moreover, we assume that only k examples from each of the C classes are available in $\mathcal{D}$, where usually $k = 1$ or $k = 5$, which is very small by neural network standards. At test time, we receive test points $\mathbf{u}^* \sim P^T(\mathbf{U})$ and our model outputs a class for the test point $\mathbf{u}^*$.

Given an image $\mathbf{u}$, our goal is to leverage both $\mathcal{D}$ and $\widetilde{\mathcal{D}}$ to learn a feature representation

$$\mathbf{X} = h(\mathbf{U})$$

of the data, as well as a mapping to class probabilities, that leads to high predictive accuracy in data drawn from the test distribution P^T. This problem setting is summarised in Figure 4.1. This chapter proposes a probabilistic framework for few-shot learning. We follow the intuition given in the introduction of the chapter and use the initial dataset $\widetilde{\mathcal{D}}$ with a two-fold objective:

i) Learn a good feature representation h for image classification using the training dataset $\widetilde{\mathcal{D}}$.

ii) Learn weights $\widetilde{\mathrm{W}}$ mapping the computed features $\mathbf{x}$ to class probabilities, and use them to regularise the weight vectors for the new classes at few-shot time.

We achieve both objectives by training a deep convolutional network on $\widetilde{\mathcal{D}}$. During few-shot learning, the top-level softmax weights corresponding to the initial $\widetilde{C}$ classes are used to regularise the weights for the new C classes, see Section 4.3.1. As a special case, we show that our method is equivalent to training an L_2 regularised logistic regression on top of the features learned in i), where the regularisation constant is set automatically using the original weights. The regularisation constant is cumbersome to estimate with methods such as cross-validation given the small number of examples available in each class.

We perform experiments on CIFAR-100 and *mini*ImageNet, where we achieve state-of-the-art results by around 6% both in 1 and 5-shot settings, see Section 4.4. Moreover, we show empirically that architectures achieving higher batch classification accuracy also achieve higher few-shot classification, further motivating the use of the initial dataset to build a good feature mapping h, and confirming empirically that the weights W in the new classes can be regularised using the initial weight matrix $\widetilde{\mathrm{W}}$. Our experiments also look closely at the problem of calibration, that is, matching the class-probabilities computed by the network to the real uncertainty about the true label, a key component for the interpretability of the probabilities given by the model. We show empirically that our framework achieves a good trade-off between calibration and accuracy. A large portion of the community focuses on meta-learning methods for few-shot learning, see Section 4.2. The high performance of our few-shot learning approach puts into context the success of these methods in problems where the features transfer well between the training task P and the few-shot distribution P^T.

4.2 Methods for Few-shot Learning

Few-shot learning has received significant attention from the community in the last few years. The success of deep learning has allowed us to perform very well in classification tasks when large amounts of labelled data are available, yet learning from few examples remains an open challenge. In this section, we present some of the methods recently introduced to address this problem. A key difference in these approaches is how the dataset $\widetilde{\mathcal{D}}$ from the training task is used during an initial phase: meta-learning methods rather *simulate* the few-shot task during training, while deep probabilistic methods use it to build features and transfer information to the parameters of the new classes, analogously to the method we propose in this chapter.

4.2.1 Meta-learning Methods

Most few-shot learning methods are trained following a meta-learning approach. In short, meta-learning shifts the focus from trying to learn one task to learning to solve tasks, in order to learn new tasks faster. Meta-learning been applied to several areas of machine learning,

including hyper-parameter optimisation and reinforcement learning (Feurer et al., 2015; Finn et al., 2017).

We refer to the first family of methods for few-shot learning as *embedding methods*. Given a small dataset $\mathcal{D}$ with C new classes, these methods embed the training points into an embedding space $\mathcal{Z}$. To compute a label at test time, the embedding of a new data point $\mathbf{u}$ is computed and the distance in some metric to the embedding of the training points is used for prediction.

It is common for few-shot learning methods using deep learning to be trained in an *episodic* fashion. In such cases, the large dataset $\widetilde{\mathcal{D}}$ is used to simulate the few-shot task during training. During an episode e, C_e classes are randomly chosen from $\widetilde{\mathcal{D}}$, and k examples from each class are randomly selected. The classes play *during training* the role of the small dataset $\mathcal{D}$. Some unseen points from the C_e chosen classes are also selected for testing. The training points from the classes in the episode are embedded using the model, the accuracy on the test points is computed, and the model parameters are updated to maximise the loss against ground truth labels.

A first example of such embedding methods are Matching Networks (Vinyals et al., 2016). Given a training sample from an episode $\{(\tilde{\mathbf{u}}_i, \tilde{y}_i)\}_{i=1}^K$, the un-weighted probabilities of the different classes for a new data point $\mathbf{u}$ are computed as

$$\hat{y} = \sum_{i=1}^{k} a(\tilde{\mathbf{u}}_i, \mathbf{u})\tilde{y}_i, \tag{4.1}$$

where $a : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a kernel measuring the similarity between the new data point and the test points. Note that the kernel a fully specifies the performance of the classifier, and its role may be seen as i) embedding points in space $\mathcal{Z}$ and ii) comparing the distances in this space. The final prediction is then a weighted average of the training labels. For simplicity, we only present one choice kernel. Let $c(\mathbf{x}, \mathbf{y}) = \mathbf{x}^t\mathbf{y}/(\|\mathbf{x}\|\|\mathbf{y}\|)$ be the cosine similarity of vectors $\mathbf{x}$ and $\mathbf{y}$. Let $f_\theta : \mathcal{X} \to \mathcal{Z}$ be an embedding function parametrised by a neural network with parameters θ. Then Matching Networks use

$$a(\tilde{\mathbf{u}}_i, \mathbf{u}) = \frac{e^{c(f_\theta(\tilde{\mathbf{u}}_i), f_\theta(\mathbf{u}))}}{\sum_{j=1}^{k} e^{c(f_\theta(\tilde{\mathbf{u}}_j), f_\theta(\mathbf{u}))}}$$

as a kernel function. The specific form of the embedding function f_θ depends on the application, as it may be a CNN for an image classification problem and a recurrent neural network for text classification.

Prototypical Networks (Snell et al., 2017) are a streamlined version of Matching Networks, and also proceed in an episodic fashion. As opposed to Matching Networks, Prototypical Networks compute a mean embedding of the training points in each episode. Given C_e classes, let S_k the the set of training points $\mathbf{u}_i$ with label k. C_e average embeddings, or

prototypes, are computed as $\mathbf{c}_k = \frac{1}{|S_k|} \sum_{i \in S_k} f_\theta(\mathbf{u}_i)$, where f_θ is once again an embedding function parametrised by a neural network with parameters θ. Let $d : \mathcal{X} \times \mathcal{X}$ be a distance function. Given a new example $\mathbf{u}$, Prototypical Networks compute the probability of class k as

$$p(y = k \mid \mathbf{u}) = \frac{e^{-d(f_\theta(\mathbf{u}), \mathbf{c_k})}}{\sum_{j=1}^{C_e} e^{-d(f_\theta(\mathbf{u}), \mathbf{c_j})}}. \tag{4.2}$$

This method is essentially assigning the new test point to the closest mean embedding in $\mathcal{Z}$ in terms of the distance function d. During training, random episodes are drawn similarly to Matching Networks, and the negative log probability in (4.2) is used as a loss function. The parameters of the network are updated via stochastic gradient descent. Other embedding methods include Siamese Networks (Koch et al., 2015), which we do not present in detail.

Recent efforts have improved the performance of Matching Networks and Prototypical Networks using data hallucination (Hariharan and Girshick, 2017; Wang et al., 2018) by training them jointly with a generative model of the data. These embedding methods learn representations for few-shot learning, but do not directly leverage concept transfer. More precisely, the nature of episodic training leads to embedding functions which are good at spreading the examples from different classes in the embedding space: the furthest different classes are embedded, the easiest and less ambiguous a new classification becomes. This does not exploit the fact that training and test classes may share a large number of features, which can be exploited for prediction. For instance, if many dog breeds are classes in an original dataset, the shared features could be used for classifying a new dog class.

A second example of meta-learning for few-shot learning is *amortised optimisation* methods (Ravi and Larochelle, 2017). These methods also consider a set of episodes constructed from the training task, or *meta-tasks*. Contrary to embedding methods, these methods consider that a neural network f_ϕ performs classification for each episode. However, this neural network is not trained via gradient descent, but the updates are dictated by a recurrent network M_θ. Given an episode e, a neural network with parameters ϕ_e is trained for classification. After a mini-batch of examples has been seen, the loss L and gradient ΔL are computed, and the updated parameters ϕ_e are computed as $M_\theta(L, \Delta L)$. At the end of the episode, θ is updated given the test accuracy of the trained network on a held-out test set. A similar method, which can be applied to a wider range of machine learning tasks, is Model-Agnostic Meta-learning (Finn et al., 2017).

Importantly, both embedding and amortised inference methods improve when the system is trained for a specific few-shot task: to perform well in 5-shot learning, training is carried out with episodes containing 5 examples in each class. The general statement appears to be that training specifically for few-shot learning is essential for building features which generalise well at few-shot testing time. The approach proposed in this chapter is more flexible; it is not tailored for a specific number of examples and, thus, does not require retraining when switching the number of few-shot examples. Moreover, Snell et al. (2017) find that using a

larger number of few-shot classes for the training episodes (e.g., train with 20 few-shot classes per episode when testing on only 5 new few-shot classes) can be beneficial, and they choose this number by cross-validation on a validation-set. This is in alignment with our finding that training with more data and more classes improves performance at few-shot time.

4.2.2 Deep Probabilistic Methods

Deep probabilistic methods include the framework introduced in this chapter. In the introduction, we motivate that our framework uses the weights of the neural network trained on the initial dataset $\widetilde{\mathcal{D}}$ to transfer information to the new classes. More precisely, given the weights of the last layer of the network for the initial classes $\widetilde{\mathrm{W}}$, we build a probabilistic model for the new weights

$$p(\mathrm{W} \mid \widetilde{\mathrm{W}}).$$

The intuition is that deep learning will learn powerful *feature representations*, whereas probabilistic inference will transfer top-down *conceptual information* from old classes. Representational learning is driven by the large number of training examples from the original classes making it amenable to standard deep learning. In contrast, the transfer of conceptual information to the new classes relies on a relatively small number of existing classes and few-shot data points, which means probabilistic inference is appropriate.

The idea of treating the weights of a model as data and build a probabilistic model on top has been applied in multi-task learning (Bakker and Heskes, 2003). The work most closely related to our own is a method for training CNNs with highly imbalanced classes (Srivastava and Salakhutdinov, 2013). Nonetheless, it shares similarities with our methods since they model the last layer weights of a neural network as a mixture of Gaussians and regularise learning the weights for the classes with few examples using Maximum a Posteriori (MAP) inference.

Burgess et al. (2016) propose an elegant approach to few-shot learning that is an instance of the framework described here: a Gaussian model is fit to the weights and inference is carried out using MAP. We present a more general framework and carry out a more thorough empirical comparison of probabilistic models and inference procedures. While not using a probabilistic approach, Qiao et al. (2018) develop a method for few-shot learning that trains a recognition model to amortise MAP inference for the softmax weights which can then be used at few-shot learning time. While this method trains the mapping from activation to weights jointly with the classifier, and thus does not learn from the weights per se, it does exploit the structure in the weights for few-shot learning.

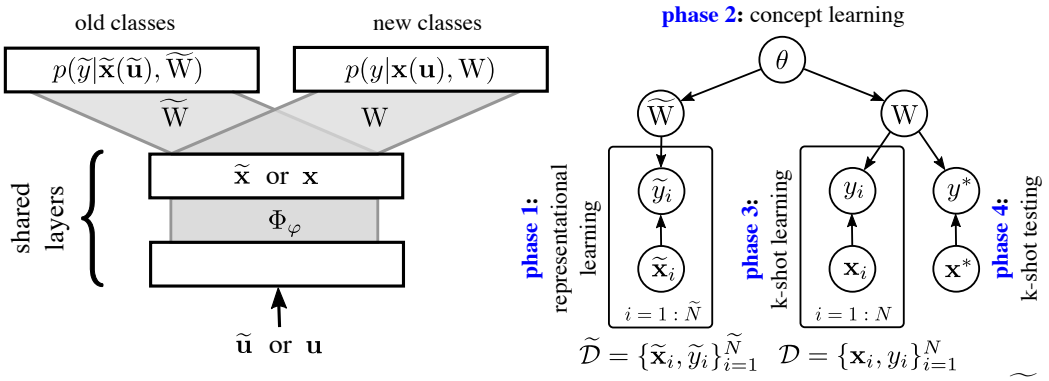


Fig. 4.2 *Left*: Shared feature extractor Φ_φ and separate top linear layers W and $\widetilde{\mathrm{W}}$ with corresponding softmax units on old and new classes. *Right:* Graphical model for probabilistic few-shot learning.

4.3 Probabilistic Few-shot Learning

Our approach to few-shot learning aims to learn a feature representation $\mathbf{X} = h(\mathbf{U}) = \Phi_\varphi(\mathbf{U})$ by training a classification model with parameters φ using the large dataset $\widetilde{\mathcal{D}}$. We transfer information to the new C classes by combining these features, which remain fixed after the initial batch classification phase, with a simple probabilistic model on the weights of the top level layer of the network. Bakker and Heskes (2003) propose a general probabilistic framework for multi-task learning in which the different tasks share the parameters up to the last layer of the network (which is essentially computing a new feature representation of the data), while the last layer weights are task-dependent. In the following, we propose a probabilistic framework for few-shot learning in this vein. Our framework comprises four phases that we refer to as 1) *representational learning*, 2) *concept learning*, 3) *k-shot learning*, and 4) *k-shot testing*, see Figure 4.2 *(right)*.

Motivation. Training a deep model on a large dataset $\widetilde{\mathcal{D}}$ provides two tools which can be used for few-shot learning. First, the activations in the last hidden layer of the network can be used as standard features for new problems. This is true for data points drawn from the training distribution, and it remains unclear to which extend these features can be used for labels which have not been observed during training. Nonetheless, one can conjecture that if the new labels share features with some of the training classes, the computed features can also be leveraged. When large shifts occur, such as observing an animal label when no animals were observed during training, several essential features may not be exploited.

Second, a classification model uses these features to compute softmax probabilities over the labels. Given the value of the features in the last hidden layer $\mathbf{x}$ for some example, the un-normalised log-probability of belonging to class j is $\mathrm{softmax}(\widetilde{\mathrm{W}}_j^t \mathbf{x})$, where $\widetilde{\mathrm{W}}_j \in \mathbb{R}^p$ is a *weight vector* for class j. This linear operation specifies how the different dimensions of the features $\mathbf{x}$ are used for classification. The main motivation for our few-shot learning methods

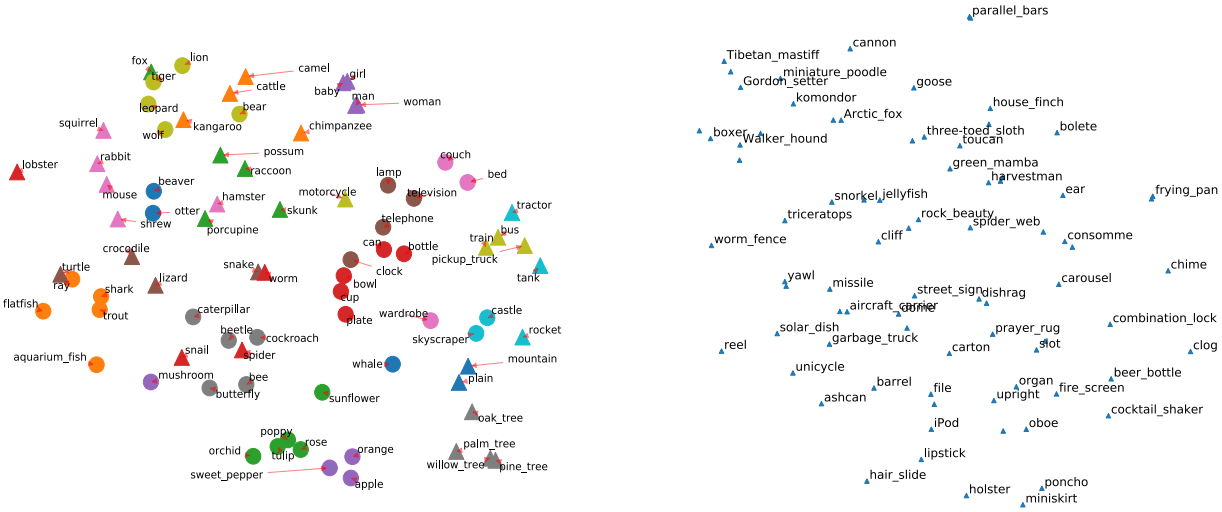


Fig. 4.3 *Left*: t-SNE embedding of the CIFAR-100 weights $\widetilde{\mathrm{W}}$ trained using a VGG style architecture. The points are coloured according to their respective superclass. The colouring by superclass makes the structure in the weights evident, as t-SNE overall recovers the structure in the dataset. For instance, oak tree, palm tree, willow tree and pine tree form a cluster on the bottom right. *Right*: t-SNE embedding of the *mini*ImageNet weights trained using a ResNet-34 architecture. Structure is still present and we observe meaningful patterns, even though the classes in *mini*ImageNet are more unique than in CIFAR-100. For instance, goose, house finch, toucan, Arctic fox, green mamba and other animals are clustered on the top, with birds close to each other. Examples of other small clusters include poncho and miniskirt, or organ and oboe. For readability, not all class names are plotted.

is that these weight vectors contain information about the classes themselves which is useful for classification of new classes. More precisely, two classes which are similar (e.g., fox and dog) exploit similar features for prediction, and their weight vectors should be closer together than for two classes with a smaller overlap (e.g., dog and table). Following this intuition, our method uses the training weights $\widetilde{\mathrm{W}} \in \mathbb{R}^{\widetilde{C} \times p}$ as data, consisting on $\widetilde{C}$ i.i.d. examples drawn from a p dimensional distribution. Given a prior distribution $P(\mathrm{W})$ on the weights, we use the training weights $\widetilde{\mathrm{W}}$ to compute the posterior $P(\mathrm{W} \mid \widetilde{\mathrm{W}})$, which we can use to regularise the weight vectors for the few-shot classes.

We embed the weight vectors trained on CIFAR-100 and *mini*ImageNet in two dimension with t-SNE, see Figure 4.3. The structure in the weight vector is apparent, as weights for similar classes are clustered together. This is particularly clear for CIFAR-100 (left), since

the clustering closely follows the super-classes defined in the dataset. In the following section, we formalize our approach to few-shot learning.

4.3.1 A Framework for Probabilistic Few-shot Learning

In this section, we present our approach for few-shot learning. In summary, the method proceeds in two steps. First, a feature extractor is trained using a large dataset, as well as weight vectors $\widetilde{\mathrm{W}}$ mapping the new obtained features to class probabilities. Following the intuition that these weight vectors $\widetilde{\mathrm{W}}$ carry information about the different classes, we fit a probabilistic model on these weights and use this model to estimate a posterior distribution for the new weight vectors W. Then, inference at test time is done by averaging the predictions over this posterior.

4.3.1.1 Learning Representations for Batch Classification

During the first phase, we use the large dataset $\widetilde{D}$ to train a classification model using standard deep learning optimisation methods. The goal of this phase is two-fold. First, we learn a rich feature representation $\mathbf{x} = \Phi_\varphi(\mathbf{u})$ from the original features. This new representation $\mathbf{x}$ is mapped to a softmax layer computing class conditional probabilities. More precisely, the feature representation $\mathbf{x} = \Phi_\varphi(\mathbf{u})$ is mapped to two sets of softmax output units corresponding to the $\widetilde{C}$ classes in the large dataset $\widetilde{\mathcal{D}}$ and the C classes in the small dataset $\mathcal{D}$, respectively. These separate mappings are parametrised by weight matrices $\widetilde{\mathrm{W}} \in \mathbb{R}^{\widetilde{C} \times p}$ for the old classes and $\mathrm{W} \in \mathbb{R}^{C \times p}$ for the new classes, where p is the size of the feature representation in the last hidden layer. The first softmax units compute $p(\widetilde{y}_n \,|\, \widetilde{\mathbf{x}}_n, \widetilde{\mathrm{W}}) = \mathrm{softmax}(\widetilde{\mathrm{W}}\widetilde{\mathbf{x}}_n)$ and the second $p(y_n \,|\, \mathbf{x}_n, \mathrm{W}) = \mathrm{softmax}(\mathrm{W}\mathbf{x}_n)$, see Figure 4.2 (left). Each row contains the weights that are associated with the corresponding class. Once the feature extractor has been trained, it is fixed for the remaining of the few-shot task. In particular, the feature extractor is not updated given new classes from the few-shot dataset $\mathcal{D}$. The softmax weights of the few-shot classes W are not trained during this phase, as the small number of examples is not fit for standard deep learning training.

4.3.1.2 Probabilistic Modelling

The next goal is to build a probabilistic method for few-shot prediction that transfers structure from the trained softmax weights $\widetilde{\mathrm{W}}$ to the new few-shot softmax weights W and combines it with the few-shot training examples. Thus, given a test image $\mathbf{u}^*$ during *few-shot testing (phase 4)*, we compute its feature representation $\mathbf{x}^* = \Phi(\mathbf{u}^*)$, and the prediction for the new label y^* is found by averaging the softmax outputs over the posterior distribution of the softmax weights given the two datasets,

$$p(y^* \,|\, \mathbf{x}^*, \mathcal{D}, \widetilde{\mathcal{D}}) = \int p(y^* \,|\, \mathbf{x}^*, \mathrm{W}) p(\mathrm{W} \,|\, \mathcal{D}, \widetilde{\mathcal{D}}) \mathrm{dW}. \tag{4.3}$$

Computing (4.3) requires two steps:

i) How do we compute the posterior of the new weights $p(\mathrm{W} \mid \mathcal{D}, \widetilde{\mathcal{D}})$ given the data?

ii) How do we compute the integral over this posterior? When (4.3) cannot be computed in closed form, an appropriate approximate inference method must be chosen.

Let us first address i) and compute the posterior $p(\mathrm{W} \mid \mathcal{D}, \widetilde{\mathcal{D}})$. We consider a general class of probabilistic models in which the two sets of softmax weights are generated from shared hyper-parameters θ, so that $p(\widetilde{\mathrm{W}}, \mathrm{W}, \theta) = p(\theta)p(\widetilde{\mathrm{W}}|\theta)p(\mathrm{W}|\theta)$ as indicated in the graphical model in Figure 4.2 (right). In this way, the large dataset $\widetilde{\mathcal{D}}$ contains information about θ that in turn constrains the new softmax weights W. The initial dataset is first used to form the posterior distribution over the hyper-parameters $p(\theta \mid \widetilde{\mathcal{D}})$. Given the few-shot dataset, we combine the information about the new weights provided by $\widetilde{\mathcal{D}}$ with the information in the few-shot dataset $\mathcal{D}$ to form the posterior distribution

$$p(\mathrm{W} \mid \mathcal{D}, \widetilde{\mathcal{D}}) \propto p(\mathrm{W} \mid \widetilde{\mathcal{D}}) \prod_n p(y_n \mid \mathbf{x}_n, \mathrm{W}) \quad \text{where} \quad p(\mathrm{W} \mid \widetilde{\mathcal{D}}) = \int p(\mathrm{W} \mid \theta) p(\theta \mid \widetilde{\mathcal{D}}) \mathrm{d}\theta. \tag{4.4}$$

To see this, notice that

$$p(\mathrm{W} \mid \mathcal{D}, \widetilde{\mathcal{D}}) \propto p(\mathrm{W}, \mathcal{D}, \widetilde{\mathcal{D}}) = p(\widetilde{\mathcal{D}}) p(W \mid \widetilde{\mathcal{D}}) p(\mathcal{D} \mid \widetilde{\mathcal{D}}, \mathrm{W}).$$

The graphical model in Figure 4.2 (right) entails that $\mathcal{D}$ is conditionally independent from $\widetilde{\mathcal{D}}$ given W, such that

$$p(\mathcal{D} \mid \widetilde{\mathcal{D}}, \mathrm{W}) = p(\mathcal{D} \mid \mathrm{W}) = \prod_n p(y_n \mid x_n, \mathrm{W}).$$

We recover (4.4) by adding $p(\widetilde{\mathcal{D}})$ to the constant of proportionality.

In order to answer i), Equation (4.4) still requires us to compute $p(\mathrm{W} \mid \widetilde{\mathcal{D}})$. We make the simplifying assumption that there is very little uncertainty on the weight vectors $\widetilde{\mathrm{W}}$ trained during phase 1, see Section 4.3.1.1, which is motivated by the large size of the training dataset $\widetilde{\mathcal{D}}$. This assumption in term implies that all the information we need to model the weights given the initial dataset is condensed in the training weights $\widetilde{\mathrm{W}}$, which we treat as observed data, and the dataset $\widetilde{\mathcal{D}}$ can thus be otherwise discarded. This results in the approximation

$$p(\mathrm{W} \mid \widetilde{D}) \approx p(\mathrm{W} \mid \widetilde{\mathrm{W}}).$$

We refer to this step as *concept learning (phase 2)* and note that all the freedom in terms of probabilistic modelling happens in the definition of $p(\widetilde{\mathrm{W}}, \mathrm{W}, \theta)$. We discuss possible models

for $p(\widetilde{\mathrm{W}}, \mathrm{W}, \theta)$ in Section 4.3.2. Finally, these approximations lead to the following for (4.4):

$$\begin{aligned} p(\mathrm{W} \,|\, \mathcal{D}, \widetilde{\mathcal{D}}) &\approx p(\mathrm{W} \,|\, \mathcal{D}, \widetilde{\mathrm{W}}) \\ &\propto p(\mathrm{W} \,|\, \widetilde{\mathrm{W}}) \prod_{n=1}^{N} p(y_n | \mathbf{x}_n, \mathrm{W}). \end{aligned} \tag{4.5}$$

Computing (4.5) corresponds to *few-shot learning (phase 3)*, where we combine the posterior given the training weights $p(\mathrm{W} \,|\, \widetilde{\mathrm{W}})$ with the few-shot dataset $\mathcal{D}$ to combine a new posterior over the weights.

Given the approximation for the posterior of the weights, we need to answer ii) in order to perform inference at test time. The posterior distribution over the new softmax weights $p(\mathrm{W} \,|\, \mathcal{D}, \widetilde{\mathrm{W}}) \propto p(\mathrm{W} \,|\, \widetilde{\mathrm{W}}) \prod_{n=1}^{N} p(y_n | \mathbf{x}_n, \mathrm{W})$ is generally intractable, and approximate inference methods must be used. This corresponds to *few-shot testing (phase 4)*.

4.3.1.3 Approximate Inference Methods for the Approximate Posterior

We present some possible approximation methods for the approximate posterior in (4.5).

A first possibility is Maximum A Posteriori (MAP) inference

$$\mathrm{W}^{\mathrm{MAP}} = \arg\max_{\mathrm{W}} p(\mathrm{W} \,|\, \mathcal{D}, \widetilde{\mathrm{W}}).$$

This optimisation problem can be solved using gradient based methods such as L-BFGS (Liu and Nocedal, 1989), since densities with respect to W of $p(\mathrm{W} \,|\, \mathcal{D}, \widetilde{\mathrm{W}})$ can be computed in closed form. In such case, the predictive integral in (4.3) collapses to one single point, and

$$\begin{aligned} p(y^* \,|\, \mathbf{x}^*, \mathcal{D}, \widetilde{\mathcal{D}}) &\approx p(y^* \,|\, \mathbf{x}^*, \mathcal{D}, \widetilde{\mathrm{W}}) \\ &\approx p(y^* \,|\, \mathbf{x}^*, \mathrm{W}^{\mathrm{MAP}}). \end{aligned} \tag{4.6}$$

As an alternative, one may consider sampling methods. The downside of considering the MAP estimate of W is that it disregards any notion of uncertainty on the weights. In principle, the small amount of labelled data from the few-shot tasks should lead to high uncertainty on the weights, which may be a downside for MAP approach. Sampling methods, such as Hybrid Monte Carlo (Neal, 2011), or HMC, return a sample $\{\mathbf{w}_i\}_{i=1}^{N}$ of N weight vectors sampled from the posterior distribution $p(\mathrm{W} \,|\, \mathcal{D}, \widetilde{\mathrm{W}})$. Then, (4.3) is computed by averaging over these samples:

$$\begin{aligned} p(y^* \,|\, \mathbf{x}^*, \mathcal{D}, \widetilde{\mathcal{D}}) &\approx p(y^* \,|\, \mathbf{x}^*, \mathcal{D}, \widetilde{\mathrm{W}}) \\ &\approx \frac{1}{N} \sum_{i=1}^{N} p(y^* \,|\, \mathbf{x}^*, \mathbf{w}_i). \end{aligned} \tag{4.7}$$

The four phases presented comprise the pipeline we propose for few-shot learning. The approximations made are a necessary step for inference, and result naturally from writing down the predictive distribution in (4.3). From a modelling perspective, the main degree of freedom remaining is the choice for the probability distribution of the weights $p(\mathrm{W})$. In the following section, we discuss different models which could be used for this purpose.

4.3.2 Choosing a Model for the Weights

The probabilistic model over the weights is key: a good model will transfer useful knowledge that improves performance. However, the usual trade-off between model complexity and learnability is particularly egregious in our setting as the weights $\widetilde{\mathrm{W}}$ are few and high-dimensional and the number of few-shot samples is small. To emphasise, each distinct class corresponds to a single "data point", and we only have $\widetilde{C}$ many data points to fit the probabilistic model; at the same time, the dimensionality of each weight vector is determined by the size of the last hidden layer, which can be very large in practice. With an eye on simplicity, we make two simplifying assumptions. First, treating the weights from the hidden layer to the softmax outputs as a vector, we assume independence. Second, we assume the distribution between the weights of old and new classes to be identical,

$$p(\widetilde{\mathrm{W}}, \mathrm{W}, \theta) = p(\theta) \prod_{c'=1}^{\widetilde{C}} p(\widetilde{\mathbf{w}}_{c'}|\theta) \prod_{c=1}^{C} p(\mathbf{w}_c|\theta) \qquad \text{where } p(\widetilde{\mathbf{w}}_{c'}|\theta) \overset{\mathrm{d}}{=} p(\mathbf{w}_c|\theta). \tag{4.8}$$

Remark 18 (Assumption of Independence of the Weights) *Assuming independence between the weight vectors may seem like a strong assumption. In practice, however, the dependency between the weight vectors is not strong. Intuitively, once the low level layers are fixed, it is reasonable to expect that a good weight vector for a given class will be aligned with the average hidden activations before the softmax layer for that class, i.e., units with high average activations should be used for prediction. Preliminary experiments show that when the softmax weights are retrained multiple times when one class is fixed and the other classes are randomly chosen, the weight vector obtained for the fixed class is always very similar. This confirms that the activations for a class are more important than which other classes are present during the optimisation.*

Approximate inference methods for the prior parameters. As a starting point, our method considers a probabilistic prior for the weights $p(\mathrm{W} \,|\, \theta)$. After observing weights $\widetilde{\mathrm{W}}$ for the $\widetilde{C}$ training classes , we compute the posterior

$$p(\mathrm{W} \,|\, \widetilde{\mathrm{W}}) = \int p(\mathrm{W} \,|\, \theta) p(\theta \,|\, \widetilde{\mathrm{W}}) d\theta. \tag{4.9}$$

This integral is often intractable, and in such cases approximate inference methods must be used. We compare the following approximation methods.

Maximum likelihood estimation (MLE) finds the parameters θ_{MLE} which maximise the likelihood of the training weights. Given the independence assumption, MLE solves the following optimisation problem:

$$\theta_{\text{MLE}} = \arg\max_{\theta} p(\widetilde{\text{W}} \,|\, \theta) = \arg\max_{\theta} \prod_{i=1}^{\widetilde{C}} p(\widetilde{\mathbf{w}}_i \,|\, \theta). \tag{4.10}$$

MLE ignores information about the prior distribution of the parameters θ. Maximum a posteriori (MAP) estimation finds instead the parameters which maximise the posterior given the data:

$$\begin{aligned} \theta_{\text{MAP}} &= \arg\max_{\theta} p(\theta \,|\, \widetilde{\text{W}}) = \arg\max_{\theta} p(\widetilde{\text{W}} \,|\, \theta) p(\theta) \\ &= \arg\max_{\theta} p(\theta) \prod_{i=1}^{\widetilde{C}} p(\widetilde{\mathbf{w}}_i \,|\, \theta). \end{aligned} \tag{4.11}$$

Both for MLE and MAP inference, the integral in (4.9) collapses to a single point.

Gaussian prior. Arguably the simplest model for the weights is a Gaussian model where $\theta = (\mu, \Sigma)$ are the mean vector and covariance matrix and $p(\mathbf{w} \,|\, \theta) = \mathcal{N}(\mathbf{w} \,|\, \mu, \Sigma)$. We choose as conjugate hyper prior the Normal-inverse-Wishart distribution $p(\theta) = p(\mu, \Sigma) = \mathcal{NIW}(\mu, \Sigma \,|\, \mu_0, \kappa_0, \Lambda_0, \nu_0) = \frac{1}{Z} |\Sigma|^{-(\nu_0+p)/2+1} e^{-\frac{1}{2} tr(\Lambda_0 \Sigma^{-1}) - \frac{\kappa_0}{2} (\mu-\mu_0)^t \Sigma^{-1} (\mu-\mu_0)}$, where Z is a normalising constant. The prior parameters are weakly data-dependent, see Murphy (2012, Chapter 4). Alternatively, the prior parameters can be chosen by cross-validation on the observed weights $\widetilde{\text{W}}$, but we found the results to be robust to changes of the hyper prior parameters. Moreover, for the covariance matrix Σ, we consider isotropic, diagonal or full covariance models in our experiments.

For the Gaussian prior, following Murphy (2012, Chapter 4), we have

$$p(\text{W} \,|\, \widetilde{\text{W}}) = \int \mathcal{N}(\text{W} \,|\, \mu, \Sigma) p(\mu, \Sigma \,|\, \widetilde{\text{W}}) \text{d}\mu \text{d}\Sigma, \tag{4.12}$$

where $p(\mu, \Sigma \,|\, \widetilde{\mathrm{W}})$ also follows a normal-inverse-Wishart distribution $\mathcal{NIW}(\mu, \Sigma \,|\, \mu_{\widetilde{N}}, \kappa_{\widetilde{N}}, \Lambda_{\widetilde{N}}, \nu_{\widetilde{N}})$ with parameters

$$\begin{aligned}
\mu_{\widetilde{N}} &= \frac{\kappa_0}{\kappa_0 + \widetilde{N}}\mu_0 + \frac{\widetilde{N}}{\kappa_0 + \widetilde{N}}\mu_{\widetilde{\mathrm{W}}} \\
\kappa_{\widetilde{N}} &= \kappa_0 + \widetilde{N} \\
\Lambda_{\widetilde{N}} &= \Lambda_0 + S + \frac{\kappa_0 \widetilde{N}}{\kappa_0 + \mu_{\widetilde{\mathrm{W}}}} \left(\mu_{\widetilde{\mathrm{W}}} - \mu_0\right)\left(\mu_{\widetilde{\mathrm{W}}} - \mu_0\right)^t \\
\nu_{\widetilde{N}} &= \nu_0 + \widetilde{N},
\end{aligned}$$

and S is the sample covariance of $\widetilde{\mathrm{W}}$ and $\mu_{\widetilde{\mathrm{W}}}$ its sample mean.

During concept learning (phase 2), we need to compute the integral in (4.12). A first solution consists on computing the MAP parameters $\theta_{\mathrm{MAP}} = (\mu_{\mathrm{MAP}}, \Sigma_{\mathrm{MAP}})$. In this case, the integral collapses and $p(\mathrm{W} \,|\, \widetilde{\mathrm{W}}) = \mathcal{N}(\mathrm{W} \,|\, \mu_{\mathrm{MAP}}, \Sigma_{\mathrm{MAP}})$.

Alternatively, (4.12) can be computed in closed form, resulting in a multivariate Student t-distribution

$$p(\mathrm{W} \,|\, \widetilde{\mathrm{W}}) = t_{\nu_{\widetilde{N}} - p + 1}\left(\mu_{\widetilde{N}}, \frac{\Lambda_{\widetilde{N}}(\kappa_{\widetilde{N}} + 1)}{\kappa_{\widetilde{N}}(\nu_{\widetilde{N}} - p + 1)}\right).$$

The single mode structure of a Gaussian distribution may be a limiting factor, since a multi-modal structure is likely in practice. A first extension is to consider mixture models.

Mixture of Gaussians (GMM) prior. A Gaussian mixture model can potentially leverage cluster structure in the weights. For instance, the weight vectors of classes corresponding to animals may be similar to each other, and quite different to the weight vectors for classes corresponding to furniture. The mixture model has parameters $\theta = (\{\Sigma_i\}_{i=1}^S, \{\mu_i\}_{i=1}^S, \{\pi_i\}_{i=1}^S)$, where S is the number of mixture components, μ_i, Σ_i are the mean and covariance of the ith Gaussian for $i \in \{1, \ldots, S\}$ and π_i is the weight of the given mode, with $\sum_{s=1}^S \pi_s = 1$. The conditional density now reads

$$p(\mathrm{W} \,|\, \widetilde{\mathrm{W}}) = \int \left(\sum_{s=1}^{S} \pi_s \mathcal{N}(\mathrm{W} \,|\, \mu_s, \Sigma_s)\right) p(\theta \,|\, \widetilde{\mathrm{W}}) \mathrm{d}\theta, \tag{4.13}$$

where $\sum_{s=1}^S \pi_s = 1$.

The integral in (4.13) is not tractable and approximate inference methods must be used. We fit the proportion parameters $\{\pi_i\}_{i=1}^S$ in two different ways. If structure in the data is known, one may use this structure to initialise the proportions. For instance, the classes in CIFAR-100 belong to 20 superclasses, with 5 classes in each of the larger groups. A possible solution is then to fit the means and covariances using MAP inference *within* each of the

clusters analogously to the Gaussian prior mentioned in the previous paragraph. A second approximation method fits the parameters of the Gaussians and the cluster allocation jointly. This can be achieved via maximum likelihood using the Expectation Maximisation (EM) algorithm (Dempster et al., 1977).

Note that, similarly to the Gaussian model, we consider isotropic, diagonal or full covariance models for the covariance matrices.

Laplace prior. As a final model, we discuss the Laplace prior. As opposed to the Gaussian models, the Laplace model encourages sparse weight vectors. This feature may be useful for our framework, as different classes may selectively rely on some of the features, and disregard others completely as they become irrelevant for classification. We thus fit a product of independent Laplace distributions, each corresponding to one of the p dimensions of the feature representation. Here, there parameters are the means and scales of the distributions, such that $\theta = (\{\lambda_i\}_{i=1}^S, \{\mu_i\}_{i=1}^S)$, and the distribution of the weights is

$$p(\widetilde{\mathrm{W}} \,|\, \{\mu_j\}, \{\lambda_j\}) = \prod_{j=1}^{p} \frac{1}{2\lambda_j} \exp\left(- \sum_{i=1}^{\widetilde{C}} \frac{|\widetilde{\mathrm{W}}_{ij} - \mu_j|}{\lambda_j} \right).$$

Maximum likelihood is used to estimate the parameters μ_j and λ_j, which lead to the following values:

$$\mu_{\mathrm{MLE},j} = \mathrm{median}_i(\widetilde{\mathrm{W}}_{ij}) \qquad\qquad \lambda_{\mathrm{MLE},j} = \frac{1}{N} \sum_i |\widetilde{\mathrm{W}}_{ij} - \mu_j|,$$

such that $p(\mathrm{W} \,|\, \widetilde{\mathrm{W}}) = \prod_j^p \frac{1}{2\lambda_{\mathrm{MLE},j}} \exp\left(- \sum_i^C \frac{|\mathrm{W}_{ij} - \mu_{\mathrm{MLE},j}|}{\lambda_{\mathrm{MLE},j}} \right)$.

For completeness, we also consider an isotropic Laplace model with mean μ and scale λ, with density function $p(\widetilde{\mathrm{W}} \,|\, \mu, \lambda) = \frac{1}{2\lambda} \exp\left(- \frac{\sum_{ij} |\widetilde{\mathrm{W}}_{ij} - \mu|}{\lambda} \right)$, where $\mu_{\mathrm{MLE}} = \mathrm{median}(\widetilde{\mathrm{W}})$ and $\lambda_{\mathrm{MLE}} = \frac{1}{Np} \sum_{ij} |\widetilde{\mathrm{W}}_{ij} - \mu|$.

Section 4.3.3 highlights the relation between a Gaussian prior on the weights and L_2 regularised logistic regression. One can similarly show that the Laplace prior is related to L_1 regularised logistic regression, which is well known for encouraging sparse weight vectors.

4.3.3 Relation to Logistic Regression

Standard logistic regression is closely related to our framework when choosing a Gaussian prior for the weights. Given a data point with features $\mathbf{x}$, logistic regression with weights W computes predictive probabilities over the labels via the softmax likelihood $p(y \,|\, \mathbf{x}, \mathrm{W}) =$

softmax(W$\mathbf{x}$). The optimal weight vector is then found via maximum likelihood:

$$W_{\text{lr}} = \arg\max_{W} \prod_{i=1}^{n} p(y_i \,|\, \mathbf{x}_i, \text{W}). \tag{4.14}$$

In practice, L_2 regularisation with constant C_{reg} is used to avoid over-fitting. In such case, the solution to the optimisation problem corresponds to the MAP solution of a model with isotropic Gaussian prior on the weights with zero mean:

$$W_{\text{lr}} = \arg\max_{W} \mathcal{N}(\text{W} \,|\, 0, \frac{1}{2C_{\text{reg}}}\text{I}) \prod_{i=1}^{n} p(y_i \,|\, \mathbf{x}_i, \text{W}). \tag{4.15}$$

This method is analogous to Equation (4.5), where $p(\text{W} \,|\, \widetilde{\text{W}}^{\text{MAP}}) = \mathcal{N}(\text{W} \,|\, 0, \frac{1}{2C_{\text{reg}}}\text{I})$. This posterior on the weights is in turn related to the MAP solution of the Gaussian model presented in Section 4.3.2. However, the probabilistic framework has several advantages. First, the modelling assumptions and approximations are made explicit, and the relation with the original dataset is clear. Second, logistic regression assumes a zero mean prior, while our method can incorporate non-zero means μ^{MAP}. Finally, the probabilistic framework provides a principled method for choosing the regularisation constant C_{reg} using the trained weights $\widetilde{\text{W}}$: $C_{\text{reg}} = 2\sigma^2_{\widetilde{\text{W}}}$, where $\sigma^2_{\widetilde{\text{W}}}$ is the empirical variance of the weights $\widetilde{\text{W}}$. Indeed, a standard technique to estimate the regularisation constant would be cross-validation with the small dataset $\mathcal{D}$. Nonetheless, few examples are available per task and cross-validation suffers, and cannot be used at all in one-shot learning.

While we do not write it explicitly, replacing the Gaussian by a Laplace prior with MAP parameters θ_{MAP} is closely related to logistic regression with L_1 regularisation. This can be a desirable modelling assumption when the weight vectors are expected to be sparse.

4.3.4 Calibration of a Classifier

Calibration is an important property of a classifier which is often overlooked. Most classifiers output a probability distribution over the output classes, and predict the class which maximises this probability. Confidence calibration looks at whether these probabilities correspond to the probability that the model is correct. Given a classifier M, consider it is possible to sample data points $\mathbf{x}$ from the data distribution for which M predicts $p(y = k \,|\, \mathbf{x}) = 0.9$ for one of the output classes k. M is calibrated if M correctly classifies these points exactly 90% of the times in the limit of infinite data. This mental experiment justifies the importance of calibration in many sensitive applications. If an autonomous vehicle outputs a probability of 1% of a pedestrian being in front of the car at a given time, we wish to interpret this as the actual probability of the pedestrian being there. In an un-calibrated classifier, this probability could in reality be significantly higher, which could have dramatic consequences.

Let M be a classifier for C distinct classes, denote by $M(\mathbf{x}) \in \mathbb{R}^C$ the probabilities assigned by the model to the C different classes, such that $\sum_{j=1}^{C} M(\mathbf{x})_j = 1$. Moreover, denote by $\hat{Y}(\mathbf{x}) = \arg\max_j M(\mathbf{x}_j)$ the class predicted by the model and Y the corresponding true label. Then, a classifier is perfectly calibrated if

$$P(\hat{Y}(\mathbf{X}) = Y \mid M(\mathbf{X}) = p) = p \quad \forall p \in [0, 1].$$

A calibration curve visualises the proportion of examples correctly classified as a function of their predicted probability; a perfectly calibrated classifier should result in a diagonal line. Similarly to Guo et al. (2017), we consider the log-likelihood of the few-shot examples and the Expected Calibration Error (ECE) as scalar measures of calibration. In the population case, we can measure the gap between confidence and accuracy as

$$\int_{p\in[0,1]} \mathbb{E}_{\mathbf{X},Y} \left[\left| P(\hat{Y}(\mathbf{X}) = Y \mid M(\mathbf{X}) = p) - p \right| \right], \tag{4.16}$$

which equals zero for a perfectly calibrated classifier.

Let us derive an estimator for (4.16) given a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$. We split the interval $[0, 1]$ in M equally sized bins, and denote by B_m the set of indexes of the data points in $\mathcal{D}$ for which the predicted probability $M(\mathbf{x})$ of the output class $\hat{Y}$ falls in the interval $]\frac{m-1}{M}, \frac{m}{M}]$. The accuracy of the classifier on B_m is $\text{acc}(B_m) = \frac{1}{|B_m|} \sum_{i\in B_m} \mathbb{1}(M(\mathbf{x}_i) = y_i)$. Similarly, the average confidence of M for examples in B_m is $\text{conf}(B_m) = \frac{1}{|B_m|} \sum_{i\in B_m} M(\mathbf{x}_i)$. The following is an estimator of (4.16)

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)|. \tag{4.17}$$

4.4 Experiments on Image Classification Tasks

We perform experiments on two standard datasets: CIFAR-100 (Krizhevsky and Hinton, 2009) and *mini*ImageNet. In phases 2 and 3 of our few-shot pipeline, different approximate inference methods and models can be used. The goal of the experiments on CIFAR-100 is to exhaustively compare these methods and analyse the implications of the approximation choice and the modelling assumptions. On *mini*ImageNet, we compare our approach against state-of-the-art methods and show that we achieve the best performance by a wide margin.

4.4.1 Model Assessment on CIFAR-100

The goal of this section is to provide an extensive model comparison of our approach on CIFAR-100. Phase 2 (concept learning) of our probabilistic framework requires both the choice of a probabilistic model for the weights $p(\mathrm{W} \mid \theta)$ and an approximate inference method

Method name	Phase 2: Concept learning	
	Prior distribution	Inference
Gauss (iso)	Gaussian isotropic covariance	MLE
Gauss (MAP prior)	Gaussian isotropic covariance	MAP
Gauss (integr. prior)	Gaussian full covariance	Integrated
GMM (supercl.)	GMM on superclasses iso. cov.	MAP
GMM (3, iso)	GMM on 3 clusters iso. cov.	MLE
GMM (3, diag)	GMM on 3 clusters diagonal cov.	MLE
GMM (10, iso)	GMM on 10 clusters iso. cov.	MLE
Laplace (diag)	Laplace diagonal covariance	MLE

Table 4.1 Description of the inference for the parameters of the prior in phase 2 (concept learning) for the models in Figure 4.4. This specifies the inference procedure for θ in $p(\mathbf{w} \mid \theta)$ after observing the training weights $\widetilde{\mathrm{W}}$.

for computing the $p(\mathrm{W} \mid \widetilde{\mathrm{W}})$, see Sections 4.3.1.3 and 4.3.2. The different models considered and corresponding inference methods can be found in Table 4.1.

During phase 4 (few-shot testing), computing the predictive distribution (4.3) is often intractable. We perform approximate inference via MAP and Hybrid Monte Carlo (HMC) sampling using NUTS (Hoffman and Gelman, 2014), see Table 4.2.

We report the log-likelihood of the weights under different models, as well as accuracy, log-likelihood and calibration in a few-shot learning task. Tables 4.1 and 4.2 describe the methods analysed for respectively phase 2 (concept learning) and phase 3 (few-shot learning) of our few-shot pipeline described in Section 4.3.1.

Method name	Phase 3: few-shot learning	
	Prior distribution	Inference
Gauss (iso) MAP	Gaussian	MAP
Gauss (MAP prior) MAP	Gaussian	MAP
Gauss (MAP prior) HMC	Gaussian	HMC
Gauss (integr. prior) MAP	Gaussian	MAP
Gauss (integr. prior) HMC	Gaussian	HMC
GMM (supercl.) MAP	GMM on superclasses	MAP
GMM (3, iso) MAP	GMM on 3 isotropic comp.	MAP
Laplace (diag) HMC	Laplace (diagonal)	HMC
Laplace (diag) MAP	Laplace (diagonal)	MAP

Table 4.2 Methods and inference procedure during phase 3 (few-shot learning) for the models used in Figure 4.4. This specifies the inference procedure used when computing $p(\mathrm{W} \mid \mathcal{D}, \widetilde{\mathrm{W}})$ for the specified prior distribution.

Model	Optimised value of mean negative log probability
Gauss (iso)	-175.9 ± 0.3
Gauss (MAP prior)	-196.1 ± 0.5
Gauss (integr. prior)	-200.6 ± 0.4
GMM 3-means (iso)	-179.0 ± 0.3
GMM 3-means (diag)	-181.2 ± 0.3
GMM 10-means iso	-181.6 ± 0.4
GMM 10-means (diag)	-181.6 ± 0.4
Laplace (iso)	-173.8 ± 0.4
Laplace (diag)	-176.6 ± 0.5

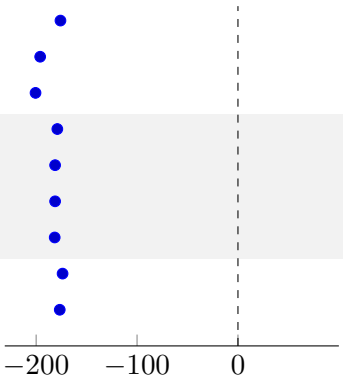


Table 4.3 Held-out log probabilities on random 70/10-splits of the training weights for the different models on CIFAR-100 (lower is better). Values are averaged over 50 splits.

Analysis of the models on held-out training weights. First, we analyse how well the chosen prior models for the new softmax weights fits the $\widetilde{C}$ training weights $\widetilde{\mathrm{W}}$. Given training weights $\widetilde{\mathrm{W}}$ and new weights $\mathrm{W} \in \mathbb{R}^{C \times p}$, the inference methods discussed (MLE, MAP and integration) allow us to compute the log likelihood of the new weights given the training weights:

$$\log\left(p(\mathrm{W} \,|\, \widetilde{\mathrm{W}})\right) = \sum_{i=1}^{C} \log\left(p(\mathbf{w}_i \,|\, \widetilde{\mathrm{W}})\right).$$

We randomly exclude 10 of the training weights $\widetilde{\mathrm{W}}$ and evaluate their held-out negative log likelihood given the remaining $\widetilde{C} - 10$ weights. This process is repeated over 50 random splits and the results of the best optimised values with respect to the hyper-parameters of the prior are shown in Table 4.3. We find that all models behave similarly but that multivariate Gaussian models generally outperform other models. We attribute the good performance of the simpler models to the small number of data points ($C - 10 = 70$ training weights) and the high dimensionality of the space, which entail that fitting even simple models is a difficult task. While more complex models may be better in the population setting, the low data regime makes more complicated models difficult to fit, leading to no improvement over the simpler models. Note that this approach constitutes a principled way to set the hyper-parameters of the prior for the weights.

Performance in the few-shot task. We compute the accuracy of our approach on a 5-way classification task for $k \in \{1, 5, 10\}$ examples per class, and average the results over 20 random splits of the 5 few-shot classes. Moreover, 10 repetitions are computed for each split with different examples seen during training.

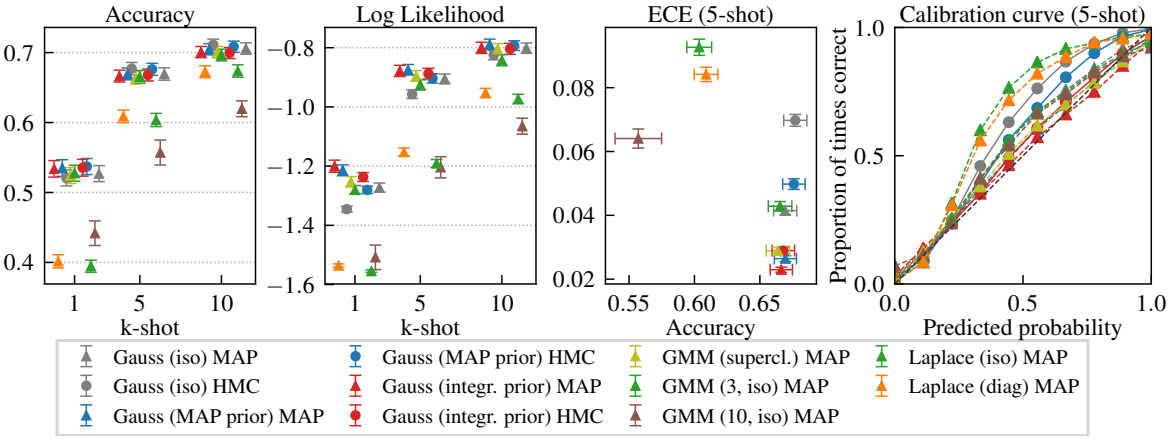


Fig. 4.4 Results on CIFAR-100 for a VGG style architecture. We report accuracy, log-likelihood and calibration for the methods and inference procedures presented in Table 4.2. With the exception of GMM (10, iso) and Laplace, all methods are similar terms of accuracy and log-likelihood. Gauss (integr. prior) HMC and Gauss (MAP) HMC are slightly better calibrated than our proposed Gauss (MAP) iso, but require significantly more computation for the sampling procedure.

Among the models considered, no statistically significant difference in accuracy is observed, with the exception of Laplace MAP and GMM (iso), which consistently underperform. These findings are consistent in terms of test log-likelihoods, see the first and second plots in Figure 4.4.

Finally, we analyse the calibration of our methods, see Section 4.3.4 for further details. Our methods are generally well calibrated, with Gaussian models generally better than Laplace models. Moreover, all methods (with the exception of Laplace and GMM (10, iso)) have low ECE (lower is better) and high accuracy, see the third and fourth plots of Figure 4.4. While Gauss (integr. prior) HMC and Gauss (MAP) HMC result in higher calibration that Gauss (MAP), we believe the gain in calibration is not worth the significant increase in computational resources needed for the sampling procedure. Interestingly, both GMM approaches are not able to outperform the other, simpler models. This is in line with the previous observation that the simpler models are better able to explain the weights. Again, we attribute this inability of mixture models to use their larger capacity to the small number of data points and the high-dimensionality of weight-space. These observations suggest that the use of mixture models in this type of few-shot learning framework is not beneficial and is in contrast to the approach of Srivastava and Salakhutdinov (2013), who employ a tree-structured mixture model. The authors compare a model in which the assignments to the superclasses in the tree are optimised over against a model with a naive initialisation of the superclass assignments, and show that the first outperforms the second. However, they do not compare against a simpler baseline such as a single Gaussian model. These conclusions may be different when the number of training classes is larger than their dimensionality.

Overall, we observe that there is no significant benefit of more complex methods over the simple isotropic Gaussian, either in terms of accuracy, log-likelihood or calibration.

Remark 19 (Recommendation for Practitioners) *The experiments carried out in CIFAR-100 suggest that using a simple Gaussian model for the weights, as well as MAP inference, performs well overall in terms of calibration and accuracy for few-shot learning. We therefore recommend that these simple models and inference schemes be used for few-shot learning to estimate all free parameters. As previously mentioned, this is true for similar scenarios in which i) the features from the training tasks transfer well to the new few-shot classes and ii) few training classes are available, thus providing few data points for the probabilistic modelling. Settings in which these assumptions are violated, e.g., adversarial classes during few-shot learning or large-scale classification problems, are left for further research.*

4.4.2 Experiments on *mini*ImageNet

*mini*ImageNet has become a standard testbed for few-shot learning and is derived from the ImageNet ILSVRC12 dataset (Russakovsky et al., 2015) by extracting 100 out of the 1000 classes. Each class contains 600 images downscaled to 84×84 pixels. We use the 100 classes (64 train, 16 validation, 20 test) proposed by Ravi and Larochelle (2017). As our approach does not require a validation set, we use both the training and validation data for the representational learning. We compare our approach against the following baselines and competing methods.

Baselines and competing methods. We compare against several baselines as well as recent state-of-the-art methods mentioned in Section 4.2. The baselines are computed on the features $\mathbf{x} = \Phi_\phi(\mathbf{u})$ from the last hidden layer of the trained CNN: (i) Nearest Neighbours with cosine distance and (ii) regularised logistic regression with regularisation constant set by cross-validation. We also compare against three recent few-shot methods: (i) Matching Networks[1] (Vinyals et al., 2016), (ii) Prototypical Networks, with numbers reported from Snell et al. (2017), (iii) Meta-learner LSTM, with numbers reported from Ravi and Larochelle (2017) and iv) Model-agnostic Meta-learning, with numbers reported from Finn et al. (2017).

Our method. Experiments on CIFAR-100 in Section 4.4.1 show that the probabilistic models we consider are comparable in terms of few-shot accuracy and performance. Therefore, we use arguably the simplest model, a Gaussian distribution with isotropic covariance $p(\mathbf{w} \,|\, \mu, \sigma^2 I) = \mathcal{N}(\mathbf{w} \,|\, \mu, \sigma^2 I)$. We estimate the parameters $\theta_{\text{MAP}} = (\mu_{\text{MAP}}, \sigma_{\text{MAP}})$ using MAP inference, see Section 4.3.2, and approximate the predictive distribution in (4.3) also via MAP inference, see Section 4.3.1.3. We note that, for the presented Gaussian model,

[1] as reimplemented and optimised by https://github.com/AntreasAntoniou/MatchingNetworks to produce results that are superior to those originally published.

the transfer is limited to a small number of parameters but already has a beneficial effect. The framework we present is general and also allows for more elaborate probabilistic models, which lead to more ambitious concept transfer, such as the Gaussian latent feature model proposed in Griffiths and Ghahramani (2011, Section 5.1).

Testing protocol. We evaluate the methods on 600 random few-shot tasks by randomly sampling 5 classes from the 20 test classes and perform 5-way few-shot learning. Following Snell et al. (2017), we use 15 randomly selected images per class for few-shot testing to compute accuracies and calibration.

Representational learning. We employ standard CNNs that are inspired by ResNet-34 (He et al., 2016) and VGG (Simonyan and Zisserman, 2014) for the representational learning on the $\widetilde{C}$ base classes, see the description of phase 1 in Section 4.3.1.

These trained networks provide both the weight vectors for the original classes $\widetilde{\mathrm{W}}$ and the fixed feature representation of the input data Φ_φ that can be exploited for few-shot learning and testing. We employed standard data augmentation from ImageNet for the representational learning but highlight that no data augmentation was used during the few-shot training and testing. For details on the architecture, training, and data augmentation see Appendix B. In addition to the full Gaussian model, we also consider the case of regularised logistic regression with regularisation constant determined by the variance of the weights, $C = 2\sigma^2_{\widetilde{W}}$, as motivated by our probabilistic framework, see Section 4.3.3. This case is commonly ignored in other works on few-shot learning and constitutes a proposed method rather than a baseline.

4.4.2.1 Analysis of the results

The experts on *mini*ImageNet lead us to the following conclusions.

Overall few-shot performance. We report performance on the *mini*ImageNet dataset in Table 4.4 and Figures 4.5 and 4.6. The best method uses as feature extractor a modified ResNet-34 with a last hidden layer of size $p = 256$, trained with all 600 examples per training class. As previously stated, we model the posterior $p(\mathrm{W} \mid \widetilde{\mathrm{W}})$ as an isotropic Gaussian for concept transfer from the original to the new classes. Despite its simplicity, our method achieves state-of-the-art and beats Prototypical Networks by a wide margin of about 6%. The baseline methods built on top of the new features $\mathbf{x} = \Phi_\varphi(\mathbf{u})$ are also state-of-the-art compared to Prototypical Networks and both logistic regressions show comparable accuracy to our methods except for on 1-shot learning. In terms of log-likelihoods, Log Reg ($C = 2\sigma^2_{\widetilde{W}}$) fares slightly better, whereas Log Reg (cv) is much worse.

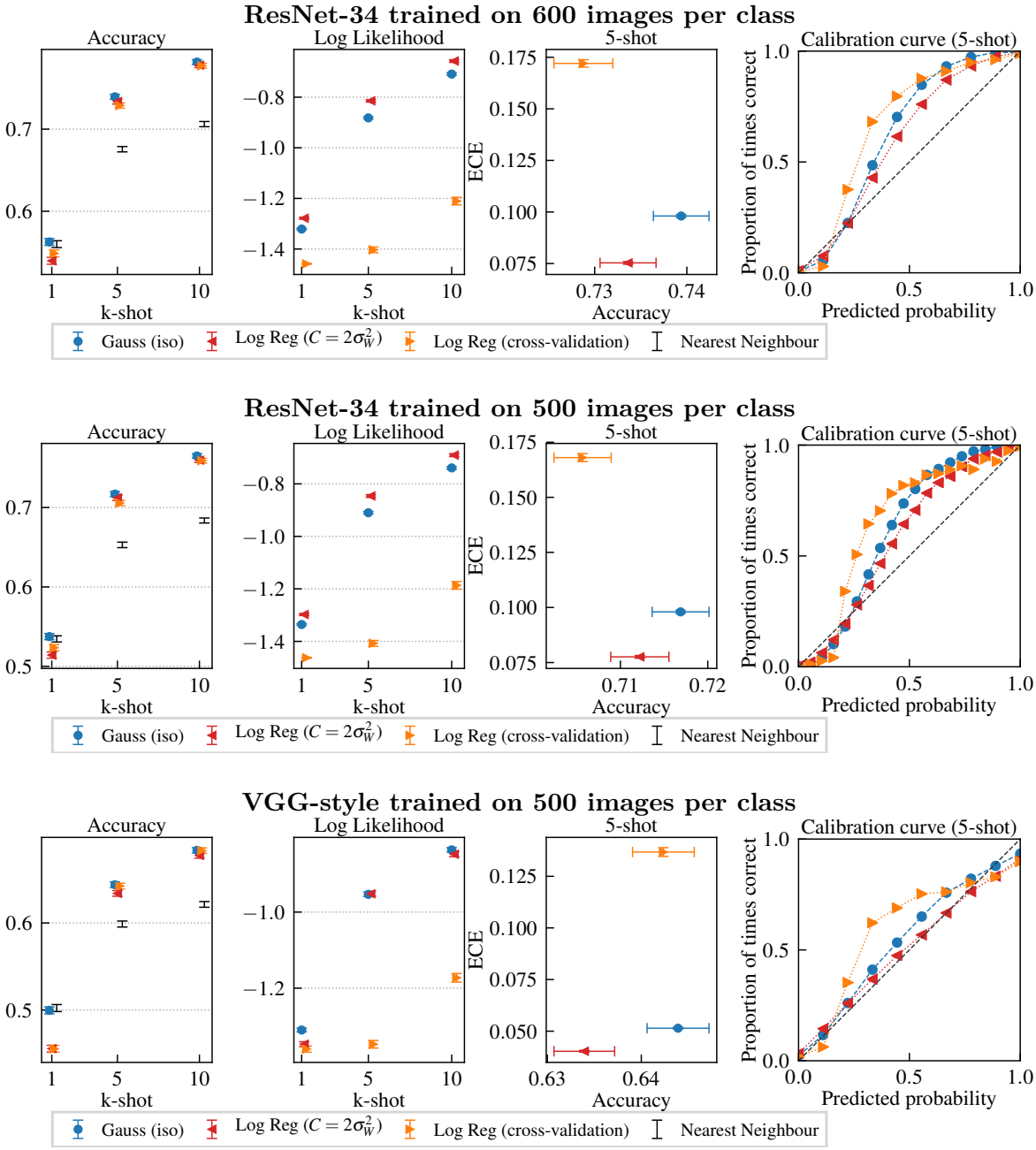


Fig. 4.5 Results for the *mini*ImageNet dataset using different network architectures and representational training. *Top:* a ResNet-34 trained with all 600 examples per class; *Middle:* a ResNet-34 trained with 500 images per class; *Bottom:* a VGG style network trained with 500 images per class. We highlight that for all three architectures the order of the different methods as well as the main messages are the same. However, the general performance in terms of accuracy and calibration differ between the architectures. The more complex architecture trained on most images performs best in terms of accuracy, indicating that it learns better features for few-shot learning. Both ResNets behave very similarly on calibration whereas the VGG-style network performs better (lower ECE and higher log likelihood as well as more diagonal calibration curve). This is in line with observations by Guo et al. (2017) that calibration of deep architectures gets worse as depth and complexity increase.

Method	1-shot	5-shot	10-shot
ResNet-34 + Isotropic Gaussian (ours)	**56.3 ± 0.4%**	**73.9 ± 0.3%**	**78.5 ± 0.3%**
Matching Networks (reimplemented, 1-shot)	46.8 ± 0.5%	-	-
Matching Networks (reimplemented, 5-shot)	-	62.7 ± 0.5%	-
Meta-Learner LSTM (Ravi and Larochelle, 2017)	43.4 ± 0.8%	60.6 ± 0.7%	-
Prototypical Nets (1-shot) (Snell et al., 2017)	49.4 ± 0.8%	65.4 ± 0.7%	-
Prototypical Nets (5-shot) (Snell et al., 2017)	45.1 ± 0.8%	68.2 ± 0.7%	-
Model-agnostic Meta-learning (Finn et al., 2017)	48.7 ± 1.84%	63.11 ± 0.7%	-

Table 4.4 Accuracy on 5-way classification on *mini*ImageNet. Our best method, an isotropic Gaussian model using ResNet-34 features consistently outperforms all competing methods by a wide margin.

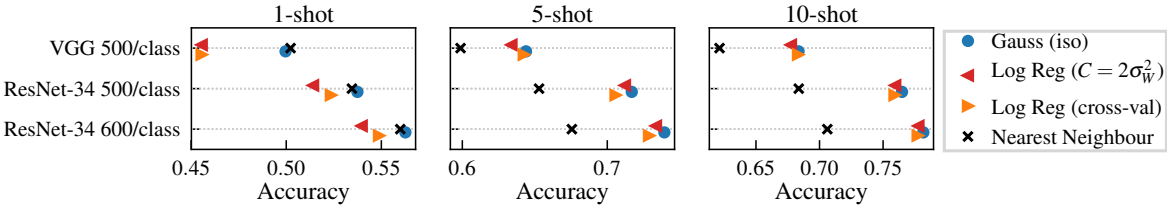


Fig. 4.6 Comparison of different network architectures and training set sizes on the few-shot learning task: VGG style network (trained on 500 images per class) and ResNet-34 style network (trained on 500 and 600 images per class, respectively). Both, deeper networks and larger number of training images, give rise to features that transfer better to few-shot learning.

Deeper features lead to better few-shot learning. We investigate the influence of different feature extractors of increasing complexity on performance in Figures 4.5 and 4.6: i) a VGG style network (500 train images per class), ii) a ResNet-34 (500 examples per class), and iii) a ResNet-34 (all 600 examples per class). We find that the complexity of the feature extractor as well as training set size consistently correlate with the accuracy at few-shot time. For instance, on 5-shot, Gauss (iso) achieves 65% accuracy with a VGG network and 74% with a ResNet trained with all available training data, a significant increase of almost 10%. Moreover, Gauss (iso) outperforms Log Reg ($C = 2\sigma^2_{\widetilde{W}}$) on 1-shot learning for all feature extractors, and performs similarly on 5- and 10-shot. We attribute the difference to the former's ability to also model the mean of the Gaussian, whereas logistic regression assumes a zero mean.

Importantly, this result implies that training specifically for few-shot learning is not necessary for achieving high generalisation performance. On the contrary, training a powerful deep feature extractor on batch classification using all of the available training data, then building a simple probabilistic model using the learned features and weights achieves state-of-the-art. Recent models that use episodic training cannot leverage such deep feature extractors as for them the depth of the model is limited by the nature of training itself. We

ran additional experiments with Matching Networks on *mini*ImageNet in which we added one or two additional convolutional layers to the architecture from the original paper. The resulting test accuracy remained unchanged.

The reference baseline in the few-shot learning literature is nearest neighbours, which performs on par with Gauss (iso) on 1-shot learning but is outperformed by all methods on 5- and 10-shot. This is evidence that building a simple classifier on top of the learned features works significantly better for few-shot learning than nearest neighbours.

Calibration. Following Guo et al. (2017), we consider the log likelihood on the few-shot test examples ECE as summary measures of calibration. Details on calibration and an introduction to the ECE can be found in Section 4.3.4. We find that Log Reg ($C = 2\sigma^2_{\widetilde{W}}$) and Gauss (iso) provide better accuracy and calibration than Log Reg (cross-validation), see Figure 4.5. The difference in calibration quality for different regularisations of logistic regression highlights the importance of choosing the right constant.

Choice of the regularisation constant for logistic regression. The results so far suggest that training a simple linear model such as regularised logistic regression might be sufficient to perform well in few-shot learning. However, while the accuracy at few-shot time does not vary dramatically as the regularisation constant changes, the calibration does, and jointly maximising both quantities is not possible. Indeed, the first two plots of Figure 4.7 show that accuracy and log-likelihood are maximised for regularisation constants which are orders of magnitude apart. The standard method to tune this constant is cross-validation, which is not applicable in the 1-shot setting, and suffers from lack of data in 5 and 10-shot. Contrary, our probabilistic framework provides a principled way of selecting this regularisation parameter by transfer from the training weights: Log Reg ($C = 2\sigma^2_{\widetilde{W}}$) strikes a good balance between accuracy and log-likelihood. The third plot in Figure 4.7 reports log-likelihood as a function of accuracy and provides further visualisation of the achieved trade-off between accuracy and calibration for Log Reg ($C = 2\sigma^2_{\widetilde{W}}$), as well as the failure of Log Reg (cross-validation) to achieve a good compromise in 5 and 10-shot.

Evaluation in an online setting. We also briefly consider the online learning setting, for which catastrophic forgetting is a well known problem (French, 1999): We jointly test on the 80 old and the 5 new classes. A Bayesian framework naturally extends to this online case, as a Bayesian posterior distribution can always be treated as a new prior to incorporate newly arriving data. During few-shot learning and testing we employ a softmax likelihood, which includes both the new and the old weights, resulting in a total of 85 weight vectors; note that we only update the weights for the new classes. We use the ResNet-34 trained on 500 images per class to retain 100 test images on the old classes for testing.

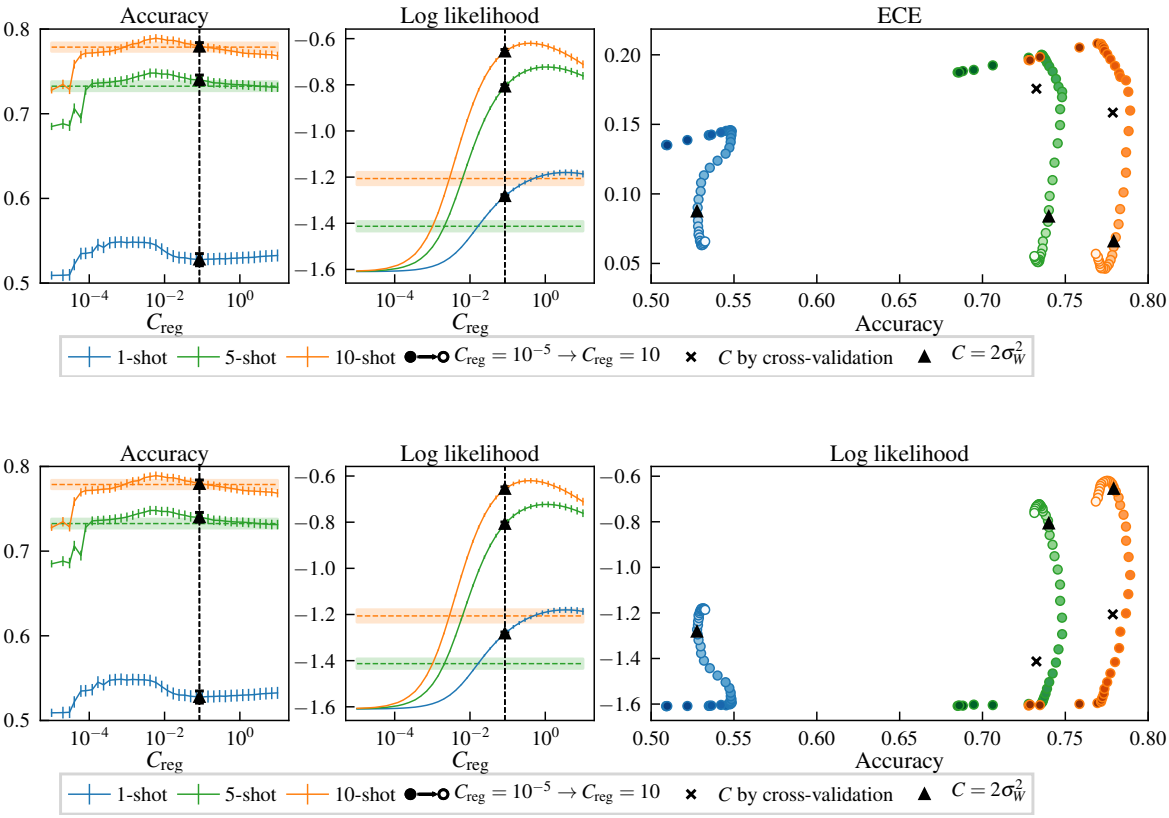


Fig. 4.7 Choice of regularisation constant for logistic regression on few-shot learning. Note that all three rows use the same raw data that are only visualised differently. *Top:* detailed plot of ECE vs. accuracy. *Bottom:* detailed plot of log likelihood vs. accuracy. Results for $C_{\text{reg}} = 2\sigma^2_{\widetilde{W}}$ are drawn as black triangles. Dashed lines correspond to logistic regression with cross-validated (changing) regularisation constant. Colour brightness of the markers ranges from dark ($C = 10^{-5}$) to bright ($C = 10$). In addition to Figure 4.7 we also provide results for calibration in terms of ECE (lower is better), which are consistent with log likelihoods (higher is better): The Bayesian inspired choice of the regularisation parameter strikes a good balance between accuracy and calibration and consistently outperforms cross-validated choice of the parameter.

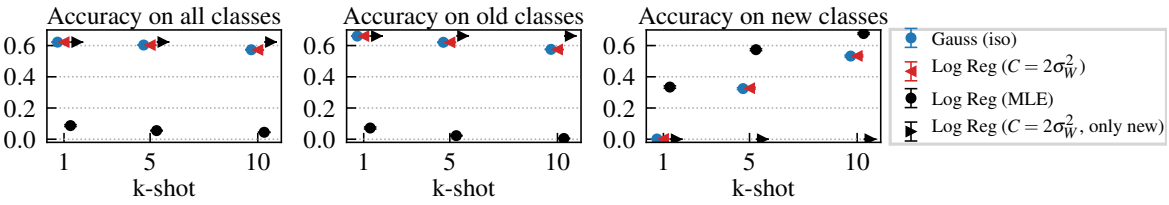


Fig. 4.8 Online learning with ResNet-34 features. Gauss (iso) and Log Reg ($2\sigma^2_{\widetilde{W}}$) strike a good trade-off between learning on new classes and forgetting of old classes. Unregularised Log Reg (MLE) and Log Reg ($2\sigma^2_{\widetilde{W}}$, only new), which has not been trained in the presence of the old weights, either completely forget the old classes or do not learn anything, respectively.

While the few-shot weights were modelled probabilistically, we fix $\widetilde{\mathrm{W}}$ for the old weights. Accuracies are reported in Figure 4.8 for i) all the 85 classes, ii) the old 80 classes only, and iii) the new 5 classes only. For 5 and 10-shot, Gauss (iso) and Log Reg ($2\sigma^2_{\widetilde{W}}$) only lose a couple of percent on the accuracy of the old classes, and perform well on the new classes, striking a good trade-off between forgetting and learning at few-shot time. For unregularised (MLE) logistic regression, the new weights completely dominate the old ones, highlighting that the right regularisation is important. Yet, cross-validation in this setting is often very challenging. When training Logistic Regression without including the old weights ("only new"), the new weights are dominated by the old ones and fail to learn the new classes, making *training in the presence of the old weights* an essential component for online learning. Our probabilistic framework automatically enforces treatment of the old weights in the likelihood as we also model them in the prior in this case.

4.5 Conclusion

This chapter introduced a framework for probabilistic few-shot learning. Fundamentally, it uses an initial large dataset to learn feature representations of the data which achieve good batch classification accuracy using standard deep learning techniques. The top-level weights of a neural network for the new, few-shot classes are regularised probabilistically using the network weights corresponding to the classes in the initial dataset. The main contribution of this chapter is to show empirically that features which achieve higher classification accuracy on the initial dataset also perform better on the few-shot learning classes. The conclusion may differ if the few-shot classes are very different from the original classes, since the features would not transfer well, but this setting is left for further research. Our findings send an important message to practitioners who need to train models for few-shot learning, as it implies that most efforts should be made in building strong features using the initial classes.

Chapter 5

Learning Representations Preserving Causal Footprints

Observational causal discovery methods for two variables, introduced in Chapter 2, search for causal footprints in a sample $\{(x_i, y_i)\} \overset{iid}{\sim} P(X, Y)$ in order to decide between $X \to Y$ and $Y \to X$. In some situations, however, we are interested in discovering the causal direction between entities for which a sample is unavailable, or a joint distribution is even ill-defined. This is the case, for instance, if we want to distinguish an original painting x from its counterfeit y, or an original document from its translated version. From the perspective of Structural Equation Models (SEMs) presented in 2.1.2, the counterfeit y is computed by applying some noisy transformation f to the original x, and this assignment is asymmetric, as differences in the original translate to a different counterfeit, but not the converse. The machinery of observational discovery can nonetheless not be applied, as causal discovery methods require a large sample from which causal footprints can be detected.

In this chapter, we propose to learn representations of the static entities x and y which *preserve* their implicit causal footprint. To accomplish this, we introduce the notion of *proxy variable*. A proxy variable W provides a source of randomness from which we can build two projections of x and y

$$A = \pi(x, W) \qquad\qquad B = \pi(y, W)$$

using a *projection function* π. The two random variables A and B are representations of x and y from which we can obtain a sample $\{(a_i, b_i)\}_{i=1}^{n} \overset{iid}{\sim} P(A, B)$ via sampling the proxy variable W. If the new features A and B preserve the same causal direction as x and y, standard causal discovery methods can be applied to these new variables to infer the direction between the static entities themselves. Section 5.1 introduces more formally the notion of proxy variable, and provides a general template for constructing the induced random features A and B. Relating back to the example of the painting x and its counterfeit y, the idea of the proxy framework is to sample several (a_i, b_i) pairs from these images using a proxy

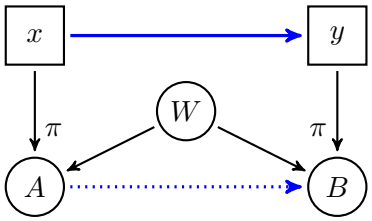


Fig. 5.1 The *static entities* (x, y) share a causal relation (thick blue arrow). A *proxy variable* W, and a *proxy projection* π produce the *random entities* (A, B), who share the causal footprint of (x, y) (dotted blue arrow). Note that no causal relation between the proxy W and the static entities x and y is assumed.

variables, and then perform causal discovery tests on these newly computed pairs. This specific example is discussed in more detail in Section 5.2.

Observational causal discovery finds causal footprints in a sample from the joint distribution of variables of interest, which is ill-defined in the case of static variables as we define them. Nonetheless, the causal relation $x \to y$ could in principle result in a measurable causal footprint, present at a different level of abstraction. The goal of the introduced proxy variables and projections is to provide a new view point on the objects x and y and to render this footprint measurable. The framework we propose is very broad and we judge its success by the ability of standard causal discovery methods to unveil the aforementioned causal footprint. Empirically, we show promising results on two areas of machine learning regarding the ability of the learned features to preserve the causal footprint of the original entities.

First, consider distinguishing cause from effect when given pairs of images x and y, where y is a transformed version of x, see Section 5.2. In particular, we show that the proposed method recovers the causal direction in over 70% of the pairs where y has been modified by a style transfer algorithm. Moreover, given shuffled frames from a video, we manage to re-order the video in time. A second application learns representations in language, see Section 5.3. There, we are given a pair of concepts such as "virus" and "death". We construct a proxy variable using a large text corpus, and show that the resulting representations allow us to recover 75% of the pairs in a dataset of causal word pairs created and scored by humans. Theoretical guarantees for the identifiability of the causal footprint between static entities x and y is left for future work.

5.1 The Concepts: Static Entities, Proxy Variables and Proxy Projections

In the following, we consider two *static entities* x, y in some space $\mathcal{S}$ that satisfy the relation "x causes y". Formally, this causal relation manifests the existence of a (possibly noisy) mechanism f such that the value y is computed as $y \leftarrow f(x)$. This asymmetric assignment

guarantees changes in the *static cause* x would lead to changes in the *static effect* y, but the converse would not hold. No other assumption is made regarding the nature of x and y.

As mentioned previously, traditional causal discovery methods cannot be directly applied to static entities. In order to discover the causal relation between the pair of static entities x and y, we introduce two main concepts: *proxy variables* W, and *proxy projections* π.

1. A *proxy random variable* W is a random variable taking values in some set $\mathcal{W}$, which can be understood as a random source of information related to x and y. Specific examples will be given in the different applications. No assumptions are made regarding the causal relation between W and the static entities.

2. A *proxy projection* is a function $\pi : \mathcal{W} \times \mathcal{S} \to \mathbb{R}$.

Given a static entity x, a proxy projection π induces a random variable $A = \pi(W, x) = \pi_x(W)$ whose randomness is driven by the proxy variable W. A sample $\{a_i\}_{i=1}^n$ thus provides n different views of the static entity x obtained by computing a projection with n values from the proxy $\{w_i\}_{i=1}^n$.

We denote by $A = \pi_x(W)$ and $B = \pi_y(W)$ the two random variables induced by the static entities x and y.

Definition 20 (Causal Proxy and Projection) *A pair of proxy variable and projection (W, π) is causal if the induced variables A and B share the same causal footprint*[1] *as the pair of static entities x and y.*

If the proxy variable and projection are causal, we may estimate the cause-effect relation between the static entities x and y in three steps.

i) Draw a sample $\{w_i\}_{i=1}^n \overset{iid}{\sim} W$.

ii) Compute the corresponding sample $\{(a_i, b_i)\}_{i=1}^n \overset{iid}{\sim} P(A, B)$.

iii) Estimate the cause-effect relation between A and B given $\{(a_i, b_i)\}_{i=1}^n$.

iv) Conclude "x causes y" if $A \to B$, or "y causes x" if $A \leftarrow B$.

The graphical model relating the components of the framework is found in Figure 5.1. A priori, no causal relation is assumed between W, x and y.

Remark 21 (Nature of the Relation Between A and B) *Chapter 2 discussed the importance of interventions when discussing causality. If X is a cause of Y, then intervening on X (by setting it to a fixed value or modifying its distribution) leads to a change in Y, while the converse is not true. The projections A and B built using the proxy variable cannot*

[1] The notion of causal footprint is introduced in Chapter 2, Section 2.2.

be causal *in the interventional sense, since performing an intervention on A has no effect on B as both A and B are constructed independently. However, a causal discovery algorithm searches for a statistically observable causal footprint which can indeed be preserved between A and B.*

Remark 22 (A Generic Form for the Proxy Variable and Projection) *We introduce the concepts of proxy variable W and causal projection π in a very general setting. The problem of establishing the causal relation between static entities is a challenging task, and we frame it deliberately in a very wide context. Any theoretical guarantees at this level are thus difficult to conceive. We believe nonetheless that keeping the framework general allows us to discuss and frame the problem well. We analyse two different problems in vision and language, for which concrete examples can be given. Since the defined proxy should preserve the causal footprint of the original static variables, we expect that applying our framework to new areas requires domain specific knowledge and extensive experimentation.*

5.2 Causal Discovery Using Proxies in Images

Consider the two images shown in Figure 5.2. The image on the left is an unprocessed photograph, while the one on the right is the same photograph after being stylised with the algorithm of Gatys et al. (2016). We represent both images x and y as a vector of pixel intensities in $\mathbb{R}^d$. From a causal point of view, the unprocessed image $x \in \mathbb{R}^d$ is the cause of the stylised image $y \in \mathbb{R}^d$. Indeed, intervening on x would lead to a different output from the style transfer algorithm, while the converse does not hold. If we wish to establish in an automatic fashion which image is the cause and which is the effect, we face a problem with two static entities x and y. How can we leverage the ideas of proxy variable and projection from Section 5.1 to recover such causal relation using a standard causal discovery method? The two degrees of freedom lie in the choice of the proxy W and the projection π.

Choice of proxy. The transformed image y is generated by modifying local patches of the original image x using a CNN, indicating that the information necessary to recover the causal direction between the two images is contained locally at the level of these patches. As such, let W be a proxy variable selecting a patch of dimensions $k \times k$ in a $d \times d$ image. More precisely, $w \sim W$ is a $d \times d$ image with ones in a random $k \times k$ patch and zeroes elsewhere.

Choice of projection. We choose as a proxy projection the standard dot product between vectors $\pi(u, v) = \langle u, v \rangle$. Given both images x and y and a sample $w \sim W$ from the proxy variable, $a = \pi(x, w)$ and $b = \pi(y, w)$ are the sums of the pixel values in the $k \times k$ patch selected by w in both images.

The following summarises the pipeline converting the two images x and y into scalar projections drawn from two random variables A and B. For $n \gg 1$ and $j = 1, \ldots, n$:

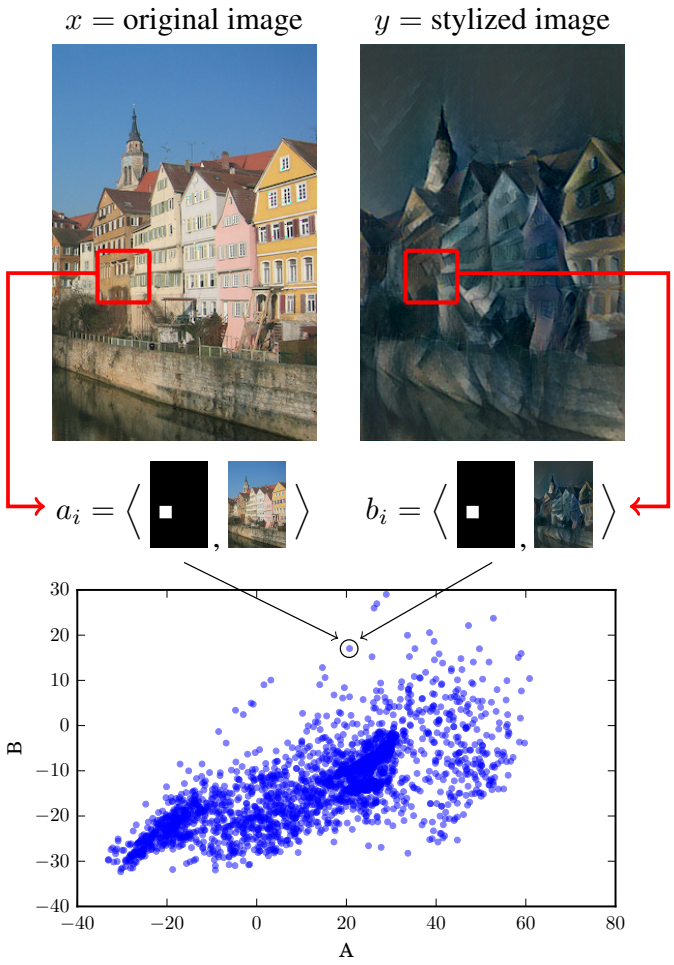


Fig. 5.2 Sampling random patches produces a proxy variable.

- Draw a mask-image w_j, which contains ones inside a patch at random coordinates, and zeroes elsewhere.
- Compute $a_j = \langle w_j, x \rangle$, and $b_j = \langle w_j, y \rangle$.

This process returns a sample $\{(a_j, b_j)\}_{j=1}^{n}$ drawn from $P(A, B)$, the joint distribution of the two scalar random variables (A, B). The conversion from static entities (x, y) to random variables (A, B) is obtained by virtue of i) the randomness generated by the *proxy variable* W, which in this particular case corresponds to random masks and ii) a causal projection π, here a simple dot product. At this point, if the causal footprint between the random entities (A, B) resembles the causal footprint between x and y, we can apply a regular causal discovery algorithm to (A, B) to estimate the causal relation between x and y. It is important to note that the specific form of W and π are relevant for this specific example, in which the modifications to image x are made *locally*. This is the case of the stylisation algorithm

of Gatys et al. (2016) and the applied filters, and explains why, in this case, selecting random patches as a proxy makes sense.

The following formalises this idea, justifying the choice of proxy and projection under some simplifying assumptions. The analysis of a fully general setting for images is left for future work.

5.2.1 Analysis in a Special Case

The intuition behind causal discovery using proxy variables is that, although we observe (x, y) as static entities, their high dimensional structure may contain footprints of the causal relation between them. If the proxy variable is designed appropriately, this information can be summarised using the projection π. But why should the causal footprint of these summaries cue the causal relation between x and y?

We formalise this question for the specific case of an original image x and its modification y, where the causal mechanism mapping x to y operates *locally* and *linearly*.

More precisely, assume that each $k \times k$ patch y_{S_i} in the stylised image is computed from the $k \times k$ patch x_{S_i} in the original image, as described by the Additive Noise Model (ANM):

$$y_{S_i} = f(x_{S_i}) + \epsilon_i,$$

where $\epsilon_i \sim \epsilon$ and ϵ is a noise variable which acts independently of the location of the subset. The size k of each subset is a property of the mechanism generating the modified image. Then, the filtered image is $y = F(x) + \epsilon$, where $F(x)_{S_i} = f(x_{S_i})$. We further assume that f is linear: $f(x_S) = \beta x_S$ where β is a $k \times k$ matrix. Then, let $P(W)$ be a distribution over masks extracting random k-subsets, and let $\pi(\cdot, \cdot) = \langle \cdot, \cdot \rangle$, to obtain:

$$\begin{aligned}
A = \pi(W, x) &= \langle W, x \rangle, \\
B = \pi(W, y) &= \langle W, y \rangle \\
&= \sum_{j,l=1}^{k} (\beta x_S)_{jl} + N \\
&= \sum_{j,l=1}^{k} \left(\sum_{u=1}^{k} \beta_{ju} x_{ul} \right) + N \\
&= \sum_{j,l=1}^{k} \alpha_j \sum_{u=1}^{k} x_{ul} + N \\
&= \left(\sum_{j=1}^{k} \alpha_j \right) \langle W, x \rangle + N
\end{aligned}$$

Experiment	**Correct causal directions detected**
ImageNet, Laplace filter	70%
ImageNet, Prewitt filter	70%
Artistic style image pairs	72%

Table 5.1 Performance of ANM on detecting the original image from its modified version. The proxy variable draws random patches from the images, and the projection computes the average pixel intensity in the patch.

where $N = \sum_{j,l=1}^{k} (\epsilon_S)_{jl}$, and where we assume that β is such that $\alpha_j = \beta_{jl}$ for all $j \leq k$. Since $A \perp N$, the pair (A, B) also follows an additive noise model which preserves the causal direction $x \to y$. We leave the investigation on identifiability conditions for causal inference using proxy variables in a more general setting to future work.

5.2.2 Numerical Experiments

We perform three experiments on images to test the efficacy of the chosen proxy to preserve the causal footprint of the original and stylised images. In these experiments, we extract $n = 1024$ square patches of size $k = 10$ pixels. Given the intuition from Section 5.2.1, we estimate the causal direction between the constructed random variables A and B using additive noise models (ANM), see Section 2.2.1 in Chapter 2. The results can be found in Table 5.1

Filters on ImageNet. We select 1000 random images from ImageNet and modify them using two transformations implemented in Scipy: Laplace and Prewitt filters. For each transformation, we evaluate our method's performance at distinguishing cause (original image) from effect (filtered image).

Artistic style modifications. We downloaded the original and stylised image pairs made using the algorithm of Gatys et al. (2016) available at https://deepart.io. This results in 2000 pairs of images. ANM recover 72% of the cause-effect relations. The results can be found in Table 5.1.

Reordering shuffled frames from a video. We decompose a video of drops of ink mixing with water into 8 frames $\{(x_i)\}_{i=1}^{8}$, shown in Figure 5.3. For each pair of images (x_i, x_j) for $i, j \in \{1, \ldots, 8\}$, we construct a scalar projection using the proxy W and projection π, and estimate the causal direction between x_i and x_j using ANM. Let M be a 8×8 symmetric matrix in which $M_{ij} = 1$ if $x_i \to x_j$ according to the ANM and $M_{ij} = 0$ otherwise. We consider M to be the adjacency matrix of the graph defined by the 8 frames, and perform a topological sort of the graph. We recover the unique true ordering among $40,320$ possibilities.

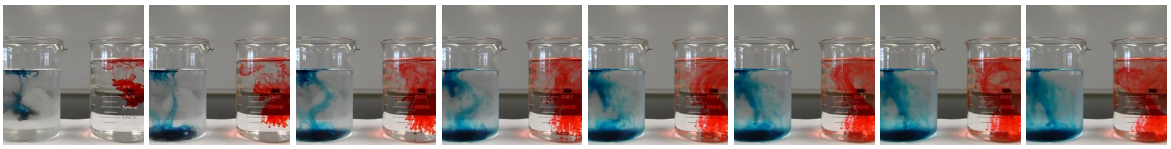


Fig. 5.3 Causal discovery using proxy variables uncovers the causal time signal to reorder a shuffled sequence of video frames. The original video can be found in http://goo.gl/sqdvu1

Remark 23 (On the Choice of Causal Discovery Algorithm) *In the example in Figure 5.2, the additive noise model assumptions are violated, since the noise is not independent from the input. Contrary to other algorithms such as RCC and NCC, ANM does not need to be trained on causal data pairs. Both RCC and NCC are introduced in Section 2.2.2 in Chapter 2. As such, better accuracy could be obtained by choosing a different causal discovery algorithm accounting for non independent noise, but it is unclear on which datasets such an algorithm should be trained. We find it encouraging that a simple approach such as ANM achieves over* 70% *accuracy in the different experiments and correctly orders the video in time.*

5.3 Causal Discovery Using Proxies in Language

In this section, we tackle the challenging problem of discovering causal relations between pairs of concepts such as "virus → death", "sun → radiation", "trial → sentence", or "drugs → addiction". We treat both words in a pair as static entities x and y, and assume that a large corpus of text $\mathcal{C}$ is available.

This problem is challenging for several reasons. First, the causal relations being established are not between words per se, but rather between higher levels concepts. The relation between them involves processes which are often complex and require significant amounts of prior knowledge. Consider for instance the pair "virus → death". We are quick in establishing that virus *causes* death since we are aware of a complex biological process underlying this causal statement: a virus infects a living organism, and by multiplying and surviving ends destroying it, which results in death. A second difficulty results form the diversity in the nature of this causal relationships. While "virus → death" refers to a biological mechanism, "trial → sentence" relates to an event and its consequence, and "musician → music" and "avocado → guacamole" are yet other manifestations of causal relations of diverse types. Finally, while a clear causal direction often appears, one can most of the times find specific contexts in which the other direction becomes causal, such as "death → virus" in *the multiple deaths in the area further spread the virus*. The causal direction we wish to discover is thus a *prototypical* causal direction, that is, the direction which seems most natural to a human in absence of a specific context. We aim to discover such causal relations by using generic observational causal discovery methods, such as the ones described in Section 2.2 of Chapter 2.

Consider the end goal of establishing the causal direction between two concepts x and y. Our first objective is to identify appropriate proxies and projections for this problem as required by Section 5.1 from both the static entities x and y and the large corpus of text $\mathcal{C}$. In order to evaluate our methods, we introduce a novel human-generated dataset of causal word pairs in Section 5.3.3, created by Amazon Mechanical Turk workers with detailed instructions. Experimental results are presented in Section 5.3.4.

First, we survey previous efforts in causal discovery in Natural Language Processing (NLP) in Section 5.3.1.

5.3.1 Causal Discovery in NLP

The NLP community has devoted much attention to the problem of identifying the semantic relation holding between two words, with causality as a special case. Girju et al. (2009) discuss the results of the large shared task on relation classification they organised (their benchmark included only 220 examples of *cause-effect*). The task required recognising relations within a given context, that is, the goal is to distinguish cause from effect given a sentence containing both words. Nonetheless, as discussed by the authors, most contexts display the default relation we are after here (e.g., "The mutant *virus* gave him a severe *flu*" instantiates the default relation in which *virus* is the cause, *flu* is the effect). All participating systems used extra resources, such as ontologies and syntactic parsing, on top of corpus data. They are thus outside the scope of the purely corpus-based methods we are considering here.

Most NLP work specifically focusing on the relation of cause and effect relies on finding words co-occurring with the target pair which are highly informative of the causal relation. A clear and obvious example is exploiting the presence of the word "because" and the position of both words relative to it in the sentence. These patterns are extracted and processed with sophisticated methods, involving annotation, ontologies, bootstrapping and/or manual filtering (see, e.g., Blanco et al. (2008); Hashimoto et al. (2012); Radinsky et al. (2012), and references therein). We experimented with extracting linking patterns from our corpus, but, due to the relatively small size of the latter, results were very sparse (note that patterns can only be extracted from sentences in which both cause and effect words occur). Recent work started looking at causal chains of events as expressed in text (see Mirza and Tonelli (2016) and references therein). Applying our generic method to this task is a direction for future work.

A semantic relation that received particular attention in NLP is that of entailment between words (*dog* entails *animal*). As causality is intuitively related to entailment, we will apply below entailment detection methods to cause/effect classification. Most lexical entailment methods rely on distributional representations of the words in the target pair. Traditionally, entailing pairs have been identified with unsupervised asymmetric similarity measures applied to distributed word representations (Geffet and Dagan, 2005; Kotlerman et al., 2010; Lenci

and Benotto, 2012; Weeds et al., 2004). We will test one of these related measures, namely, *Weeds Precision (WS)*. More recently, Santus et al. (2014) showed that the relative entropy of distributed vectors representing the words in a pair is an effective cue to which word is entailing the other, and we also look at entropy for our task. However, the most effective method to detect entailment is to apply a supervised classifier to the concatenation of the vectors representing the words in a pair (Baroni et al., 2012; Roller et al., 2014; Weeds et al., 2014).

5.3.2 Static Entities, Proxies, and Projections for NLP

In the language of causal discovery with proxy variables, a pair of concepts denoted by words is a pair of static entities: $(x, y) = (\text{virus}, \text{death})$. In order to discover the causal relation between x and y, we need to design a proxy variable W and a projection function π.

Our intuition for this step relies on several observations. First, much can be said about a word by its context, i.e., the words which appear next to it in a large corpus. This is often referred to as the Distributional Hypothesis (Sahlgren, 2008). As such, we construct a proxy which samples words from the corpus in a representative fashion. Moreover, word2vec representations (Mikolov et al., 2013) are known to preserve information about the similarity between words and the likelihood of words co-occurring in a corpus. We thus design projections based on different comparisons of the embeddings of the words in question.

As a proxy, we denote by W a random variable sampling words from the vocabulary from which the large corpus $\mathcal{C}$ is sampled. Moreover, the distribution of W is simply a unigram distribution on $\mathcal{C}$, that is, the frequency distribution of the words in the corpus. Words such as "the" or "because" will have a large probability mass, while rare words such as "guacamole" have a smaller mass.

Given two static entities x and y and a word $w \sim W$, we consider several projections π. Mostly, these are based on word2vec representations of v_x, v_y and v_w, such that $\pi(x, w) = f(v_x, v_w)$ and $\pi(y, w) = f(v_y, v_w)$ for some scalar function f.[2] A detailed description of the projections is given in Section 5.3.2.1.

Once we have defined the causal projection π, we can proceed in the following way:

i) Sample words from the proxy $w_1, \ldots, w_n \sim P(W)$. Specifically, we estimate $P(W)$ from a large corpus of natural language, and sample $n = 10,000$ words without replacement.[3]

ii) Compute the projections $a_i = \pi(w_i, x)$ and $b_i = \pi(w_i, y)$ for $i \in \{1, \ldots, n\}$.

[2]The nature of both the proxy and the projection differs greatly from the ones used in the vision experiments in Section 5.2. This reinforces the point that the framework of proxy variables is *flexible* and requires *domain-specific* knowledge, since no fit-for-all solution exists.

[3]This is equivalent to sampling approximately the top 10,000 most frequent words in the corpus. Due to the extremely skewed nature of word frequency distributions (Baayen, 2001), sampling with replacement would produce a list of very frequent words such as *a* and *the*, sampled many times.

iii) Estimate the causal direction between A and B given the sample $\{(a_i, b_i)\}_{i=1}^n$ using a causal discovery algorithm.

5.3.2.1 Causal projections

Throughout our experimental evaluation, we will use and compare different proxy projections. Denote by $v_x^i \in \mathbb{R}^d$ and $v_x^o \in \mathbb{R}^d$ the input and output word2vec representations respectively of word x.

1) $\pi_{\text{w2vii}}(w, x) = \langle v_w^i, v_x^i \rangle$. The dot-product $\langle v_w^i, v_x^i \rangle$ measures the *similarity in meaning* between w and x. Therefore, the dot product of the input embeddings of two very similar words such as "cat and dog" or "cat and auto" will be large, while the dot product is small for distant words such as "dog and pipe".

2) $\pi_{\text{w2vio}}(w, x) = \langle v_w^i, v_x^o \rangle$. The dot-product $\langle v_w^i, v_x^o \rangle$ is an un-normalized estimate of the conditional probability $p(x \mid w)$ (Melamud et al., 2015). Therefore, this projection is more indicative of words that co-occur often in the corpus. For instance, regardless of the similarity between the words, the dot product of the output embeddings of "dog and food" should be high, while "because and thus" are often exclusive and thus lead to a small dot product.

3) $\pi_{\text{w2voi}}(w, x) = \langle v_w^o, v_x^i \rangle$, an un-normalized estimate of the conditional probability $p(w \mid x)$.

4) $\pi_{\text{counts}}(w, x) = p(w, x)$, where the joint probability $p(w, x)$ is directly estimated from counting within-sentence co-occurrences in the corpus.

5) $\pi_{\text{prec-counts}}(w, x)$ similar to the one above, but only over sentences where w precedes x.

6) $\pi_{\text{pmi}}(w, x) = p(w, x)/(p(w)p(x))$, where the probability distributions $p(w)$, $p(x)$, and $p(w, x)$ are estimated from counting words and (sentence-based) co-occurrences in the corpus. The log of this quantity is known as point-wise mutual information, or PMI (Church and Hanks, 1990).

7) $\pi_{\text{prec-pmi}}(w, x)$, similar to the one above, but only over sentences where w precedes x.

Applying the causal projections to our sample $\{w_i\}_{i=1}^n$ from proxy W, we construct the n-vector

$$\Pi_{\text{proj}}^x = (\pi_{\text{proj}}(w_1, x), \ldots, \pi_{\text{proj}}(w_n, x)), \tag{5.1}$$

and similarly for Π_{proj}^y, where

$$\text{proj} \in \{\text{w2vii}, \text{w2vio}, \text{w2voi}, \text{counts}, \text{prec-counts}, \text{pmi}, \text{prec-pmi.}\}. \tag{5.2}$$

Π^x_{proj} is the vector of projections $(a_1, \ldots, a_n)$, and $\Pi^y_{\text{proj}} = (b_1, \ldots, b_n)$. We use the skip-gram model of fastText (Bojanowski et al., 2017) to compute 300−dimensional word2vec representations.

5.3.3 A Real-World Dataset of Cause-Effect Words

We introduce a human-elicited, human-filtered dataset of $10,000$ pairs of words with a known causal relation. This dataset was constructed using Amazon Mechanical Turk (AMT) in two steps[4].

The first step is a *creation* step. AMT workers are given clear examples of word pairs sharing a causal relation (e.g., "sun causes radiation"), and importantly words which are simply associated but not causal (e.g., "knife" and "fork"). Moreover, the AMT workers have a monetary incentive to create diverse word pairs. The detailed instructions given to the workers are reported in Appendix C. This step results in $n = 10,000$ word pairs $\{(x_i, y_i)\}_{i=1}^n$.

The second step is a *voting* step. Each of the pairs collected is shuffled and submitted to 20 AMT workers which did not participate in the creation step. Each worker was asked to vote whether a pair of words (x, y) satisfies "x causes y", "y causes x", or "x and y do not share a causal relation". For more details, see Appendix C.2.

Both steps carried out by the AMT workers result in a dataset of $10,000$ word pairs, each accompanied by three numbers: the number of turks that voted "x causes y", the number of turks that voted "y causes x", and the number of turks that voted "x and y do not share a causal relation".

For our experiments below, we only consider the pairs for which at least 18 out of 20 AMT workers voted that $x \to y$. This results in $N = 1,970$ word pairs. We do this selection for two reasons. First, we want to be confident that the pairs considered are causal, and worker confidence is taken as a measure of the prototypical causal direction we are interested in discovering (that is, the strongest causal link as judged by a human in absence of a specific context). Indeed, several of the obtained pairs are simply associated, such as "awful and gross", which is voted as associated by 80% of the workers. Moreover, given strong confidence from the AMT workers, we can consider that the problem is *binary*. Therefore, our algorithm must only output "$x \to y$" or $y \to x$, and does not consider the possibility of a confounded pair.

5.3.4 Experiments

We evaluate a variety of methods to discover the causal relation between two words appearing in a large corpus of natural language. We study methods that fall within three categories: *baselines*, *distribution-based causal discovery methods*, and *feature-based supervised methods*. These three families of methods consider an increasing amount of information about the

[4]The dataset can be found in https://github.com/facebookresearch/CausalityProxyVariables

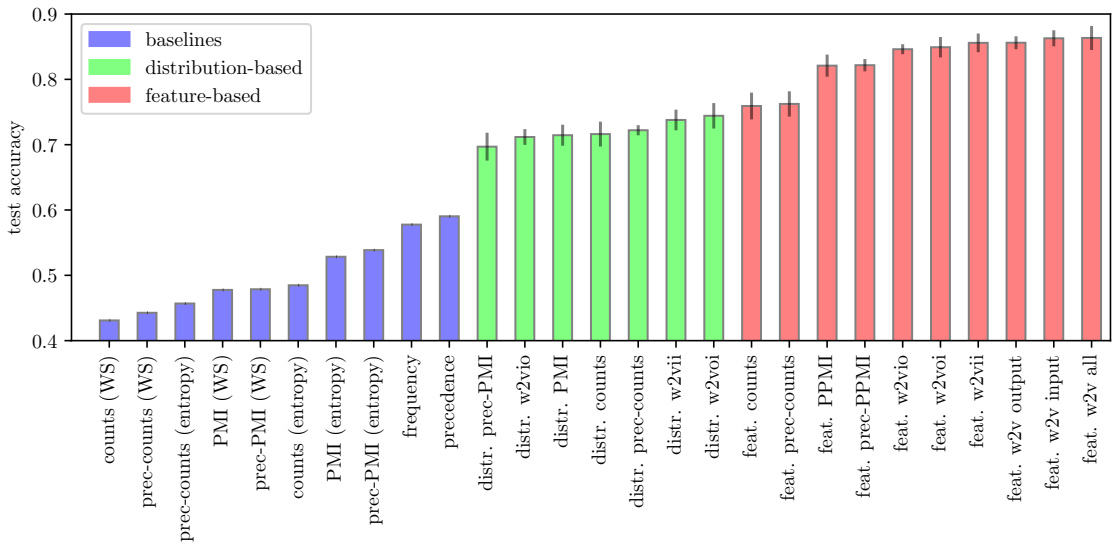


Fig. 5.4 Results for all methods on the NLP experiment. Accuracies above 52% are statistically significant with respect to a Binomial test at a significance level $\alpha = 0.05$.

task at hand, and therefore exhibit an increasing performance up to 85% classification accuracy. Our computations are based on the full English Wikipedia, as post-processed by Matt Mahoney (see http://www.mattmahoney.net/dc/textdata.html). We now present in more detail the three families of methods we implement.

5.3.4.1 Baselines

We consider a variety of unsupervised baselines, each of which computes two scores $S_{x\to y}$ and $S_{x\leftarrow y}$ for both possible causal directions. A baseline then predicts $x \to y$ if $S_{x\to y} > S_{x\leftarrow y}$, and $x \leftarrow y$ if $S_{x\to y} < S_{x\leftarrow y}$.

We now introduce the different baselines with their corresponding scores $S_{x\to y}$and $S_{x\leftarrow y}$:

- *frequency*: $S_{x\to y}$ is the number of sentences where x appears in the corpus $\mathcal{C}$, and $S_{x\leftarrow y}$ is the number of sentences where y appears in the corpus. *frequency* decides that $x \to y$ if x is more frequent than y in $\mathcal{C}$.

- *precedence*: considering only sentences from the corpus where both x and y appear, $S_{x\to y}$ is the number of sentences where x occurs before y, and $S_{x\leftarrow y}$ is the number of sentences where y occurs before x. *precedence* implicitly assumes that the cause often precedes the effect in a given sentence.

- *counts (entropy)*: $S_{x\to y}$ is the entropy of Π^x_{counts}, and $S_{x\leftarrow y}$ is the entropy of Π^y_{counts}, as defined in (5.1).

- *counts (WS)*: Using the WS measure of Weeds and Weir (2003), $S_{x\to y} = \mathrm{WS}(\Pi^x_{\text{counts}}, \Pi^y_{\text{counts}})$, and $S_{x\leftarrow y} = \mathrm{WS}(\Pi^y_{\text{counts}}, \Pi^x_{\text{counts}})$.
- *prec-counts (entropy)*: $S_{x\to y}$ is the entropy of $\Pi^x_{\text{prec-counts}}$, and $S_{x\leftarrow y}$ is the entropy of $\Pi^y_{\text{prec-counts}}$.
- *prec-counts (WS)*: analogous to the previous.

The baselines *PMI (entropy)*, *PMI (WS)*, *prec-PMI (entropy)*, *prec-PMI (WS)* are analogous to the last four, but use $(\Pi^x_{\text{(prec-)pmi}}, \Pi^y_{\text{(prec-)pmi}})$ instead of $(\Pi^x_{\text{(prec-)counts}}, \Pi^y_{\text{(prec-)counts}})$. Figure 5.4 shows the performance of these baselines in blue.

5.3.4.2 Distribution-based Causal Discovery Methods

The methods in this section implement our framework of causal discovery using proxy variables. They classify n samples from a 2-dimensional *probability distribution* as a whole. Recall that a vocabulary $\{w_j\}_{j=1}^n$ drawn from the proxy is available. Given N word pairs (x_i, y_i), this family of methods constructs a dataset

$$\mathcal{D} = \left\{ (\{(a^i_j, b^i_j)\}_{j=1}^n, \ell^i) \right\}_{i=1}^N ,$$

where $a^i_j = \pi_{\text{proj}}(w_j, x_i)$, $b^i_j = \pi_{\text{proj}}(w_j, y_i)$, $\ell^i = +1$ if $x_i \to y_i$ and $\ell^i = -1$ otherwise. In short, $\mathcal{D}$ is a dataset of N "scatterplots" annotated with binary labels. The i-th scatterplot contains n 2-dimensional points, which are obtained by applying the causal projection to both x_i and y_i, against the n vocabulary words from the proxy.

The samples $\{(a^i_j, b^i_j)\}_{j=1}^n$ are computed using a deterministic projection of iid draws from the proxy, meaning that $\{(a^i_j, b^i_j)\}_{j=1}^n \sim P^n(A^i, B^i)$. Therefore, these methods search for causal footprints *at the 2-dimensional distribution level*, and we term them *distribution-based causal discovery methods*. The methods in this family first split the dataset $\mathcal{D}$ into a training set $\mathcal{D}_{\text{tr}}$ and a test set $\mathcal{D}_{\text{te}}$. Then, the methods train RCC on the training set $\mathcal{D}_{\text{tr}}$, and test its classification accuracy on $\mathcal{D}_{\text{te}}$. This process is repeated ten times, splitting at random D into a training set containing 75% of the pairs, and a test set containing 25% of the pairs. Each method builds on top of a causal projection from (5.2) above. Figure 5.4 shows the test accuracy of these methods in green.

5.3.4.3 Feature-based Supervised Methods for an Upper-bound on Test Performance

Similarly to successful methods for other NLP applications such as entailment, we consider treating this as a pure classification problem. More precisely, given the feature vectors in (5.1)

and N word pairs (x_i, y_i), we create a dataset

$$\mathcal{D} = \{(u^i, \ell^i)\}_{i=1}^N,$$

where $\ell^i = +1$ if $x_i \to y_i$, $\ell^i = -1$ if $x_i \leftarrow y_i$, and "proj" is a projection from (5.2), and $u^i = (\Pi_{\text{proj}}^{x_i}, \Pi_{\text{proj}}^{y_i})$ is a $2n$ dimensional feature vector obtain by concatenating the vectors in (5.1) for x and y.

Next, we split the dataset $\mathcal{D}$ into a training set $\mathcal{D}_{\text{tr}}$ containing 75% of the pairs, and a disjoint test set $\mathcal{D}_{\text{te}}$ containing 25% of the pairs. To evaluate the accuracy of each method in this family, we train a random forest of 500 trees using $\mathcal{D}_{\text{tr}}$, and report its classification accuracy over $\mathcal{D}_{\text{te}}$. This process is repeated ten times at random. The results are presented as red bars in Figure 5.4. We also show the classification accuracy of training the random forest on the raw word2vec representations of the pair of words (top three bars).

5.3.4.4 Discussion of the Results

The results from the three families of methods can be found in Figure 5.4. Some of the baselines (in blue) obtain statistically significant results. The best performing baseline at 59% is precedence. We believe its success can be explained by the fact that the English language is more often written in an active voice, and this baseline may perform badly on another language. Frequency also performs above chance. This may be due to a bias in the dataset, or simply to the strong relation between causality and entailment. It is likely that a cause word leads to many possible effects, and thus that it appears more frequently. The other baselines do not achieve above chance performance.

Distribution-based methods (in green) all achieve above chance performance, with w2vii and w2voi performing best at 75% accuracy. Pairs are classified based on distributional features of the sample $\{(a_i, b_i)\}_{i=1}^n$, and are thus robust to permutations and removals of the samples. In our experiments, we use a fraction of the word pairs to fit RCC. Further research could attempt to use a black-box causal discovery algorithm to classify the pairs, an extra guarantee that the distributional signal used for classification is purely causal. Moreover, this would allow for each pair to be classified independently of the size of the word pair dataset. Nonetheless, developing a general classifier for causal discovery is still an open area of research.

The feature-based methods (in red) perform best, with w2v input and w2v output reaching up to 85% test classification accuracy. While we believe this result is impressive, it is not possible to establish which signals are being exploited for classification, and in particular whether the classifier has discovered a causal signal. Moreover, feature-based methods *require* a labelled dataset of causal word pairs, and similarly to any i.i.d. supervised learning problem, their performance heavily relies on the size of such dataset. We thus consider the performance of this family of methods as an upper-bound on achievable test performance for this dataset.

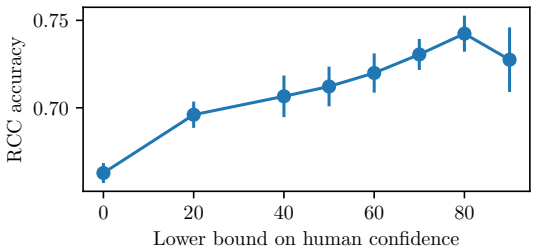


Fig. 5.5 RCC accuracy versus human confidence. The accuracy of RCC improves jointly with the confidence of the human annotators, further evidence that our methods captures a notion of the causal direction between concepts aligned with the one of the annotators.

5.3.4.5 Relation between RCC and human annotators

Out of the $10,000$ pairs, we compute the classification accuracy of distribution-based methods successively on the pairs achieving over $k\%$ agreement between the AMT workers, for $k \in \{0, 20, 40, 50, 60, 70, 80, 90\}$. For the small values of k, many of the pairs considered are confounded or display an inverse causal direction. For larger values, however, only pairs which align with the human notion of causality are considered. The performance of RCC is strongly correlated with the notion of causality from human annotators, see Figure 5.5. We consider this an encouraging sign that our proxy variables and projections arguably capture a notion of causality aligned with the one of human annotators.

5.4 Conclusion and Thoughts for Proxy Variables in Broader Machine Learning

This chapter framed the problem of causal discovery between static entities x and y, and introduced the notions of proxy variable W and proxy projection π. The framework proceeds as follows: a sample from the proxy variable is used to compute a sample $\{(a_i, b_i)\}_{i=1}^n$, on which a causal footprint can be discovered by a causal discovery algorithm. We focused this general framework into two specific problems.

First, we showed that we can identify an image from its modification by a variety of filters and a style transfer algorithm. Moreover, we showed on one example that our algorithm is able to order shuffled video frames into the correct time ordering. Second, we study an exhaustive family of baselines and projections when establishing the causal relation between pairs of concepts. In particular, we introduced a dataset of $10,000$ word pairs together with human confidence on the correct causal direction, and manage to correctly classify up to 75% of the pairs.

Several exciting research questions remain open. First, obtaining theoretical guarantees on the identifiability of the causal direction between x and y in specific examples remains a challenging question, as well as constructing more systematic protocols for choosing proxies and projections. It would be particularly interesting if we could establish conditions on W and π such that the causal direction between x and y is preserved in A and B, as well as conditions which render the causal direction not identifiable analogously to the bivariate Gaussian case for standard causal discovery. Finally and more broadly, the notion of proxy variable could be exploited in other contexts in broader machine learning. For instance, consider a supervised learning problem mapping a *feature* random variable $\mathbf{X}$ into a *target* random variable Y. Such problem is often solved by considering a sample $\{(\mathbf{x}_i, y_i)\}_{i=1}^n \sim P^n(\mathbf{X}, Y)$. In this scenario, we may contemplate an unpaired, external, *proxy* source of information w (such as a memory), which might help solving the supervised learning problem at hand. One could incorporate the information in the static source w by constructing the proxy projection $w_i = \pi(\mathbf{x}_i, w)$, and add them to the dataset to obtain $\{((\mathbf{x}_i, w_i), y_i)\}_{i=1}^n$ to build the predictor $f(\mathbf{x}, \pi(\mathbf{x}, w))$.

Chapter 6

Discussion and Conclusion

This thesis introduced methods for building transferable representations for different areas of machine learning. These include invariant feature mappings for distribution shift problems, batch-based features for transfer to few-shot classes, and projections which preserve the causal footprint between two static entities. We discuss some of the insights gained, as well as future directions.

6.1 Causality as a Language for Transfer

This thesis motivates the use of causality as a natural language for transfer. When humans learn to interact in the world, causal laws are a stable source of knowledge from which we can transfer. Our knowledge of gravity is learned from interacting with the world and observing how objects react to these interactions (Battaglia et al., 2013). When given the first frames of a video in which a structure is collapsing, we are able to build a mental model of the object and predict which part of the object lands in different parts of space. Our knowledge of these invariants in the world is built on top of repeated observation of outcomes, and can be re-used in different configurations of the world.

This motivates us to frame distribution shift under assumptions inspired by causality. We assume that all training tasks, as well as the test task, are drawn from joint distributions induced by the same causal graph. In a prediction problem with features $\mathbf{X}$ and target Y, we assume that a subset of the features S^* leads to *invariant conditionals*

$$Y \mid \mathbf{X}_{S^*}.$$

This conditional distribution represents a mechanism which is invariant between tasks and corresponds to a fundamental causal property of the world.

We showed that using only features in an invariant set S^* is optimal when the test task is chosen adversarially. This notion of invariance could be useful in other areas of machine

learning. The message implied in these results is that *exploiting statistical dependencies for prediction should be performed with care when the distribution of the data changes.* This is clear in problems such as reinforcement learning (Kansky et al., 2017). For instance, an agent achieving very high scores on Atari games performs poorly when the background colour of the image changes. This seemingly insignificant change modifies the joint distribution of the input data and without any further assumptions, there is no reason to expect that the agent should generalise to this new scenario. Two tasks with two background colours seem to be very close in some metric, yet very distant in another one: all the *principles* and *mechanisms* valid in one are valid in the other, yet the *statistics* of the input data differ. In order to build agents which generalise to radically new settings, we expect that encouraging learning invariant mechanisms, rather than only good features for i.i.d. prediction, should help.

Clearly, in most machine learning problems, a subset of the features does not cause the target. It is plausible, nonetheless, that we can extend the assumption of this paper to the existence of a feature mapping h^* such that

$$Y \mid h^*(\mathbf{X})$$

is invariant between the different tasks. An invariant mechanism can thus be learned at the level of the new features $h^*(\mathbf{X})$.

Two interesting directions for further research are the extension of our assumptions to classification and reinforcement learning problems. Classification leads to some difficulties in terms of the assumptions themselves: how should be frame the invariance $Y \mid \mathbf{X}_{S^*}$ in classification? Most classification problems are actually *anticausal* (Schölkopf et al., 2012): in handwritten digit classification, the digit is the output of the guided intent of the author, and therefore the label or intent cannot be written as a function of the hand written digit. The label we predict on an input, nonetheless, is causal: intervening on the input image may make us change our impression on the true label of the digit (for instance, adding a horizontal line on top of a four rather looks like a nine). One could conceive a training protocol in which an agent intervenes on the high level features of a classification network, and is rewarded by how badly the network predicts. As long as the agent has a way to guarantee that the intervened example belongs to the same class (for instance, it's still a cat but with a very off domain background), the network should be encouraged to learn features which are truly predictive of a cat, and thus generalise to a larger image distribution.

Experimentation is at the core of reinforcement learning, and we believe this is the area in which the invariance properties discussed could have the largest playing ground.

6.2 Modularity of Causal Mechanisms and Assumptions

A key conclusion from Chapter 3 is that causal assumptions lead to modular, re-usable models which are robust to distribution changes. We showed that the experts trained on MNIST generalise to a new distribution, unseen during training. We believe this generalisation is due to the training protocol, since the fully convolutional nature of the experts and the initialisation to the identity in the transformed digits encourage generalisation. Certainly, we need extensive experimentation regarding properties of the distributions in which the experts could be readily deployed. Omniglot remains a single channel dataset with relatively simple symbols, and the experts would most likely fail in a more complex dataset such as ImageNet. Nonetheless, the evidence suggests that the model truly learned the *functions* mapping the digits back to their counterpart. Since this is an image dataset, we can visually assess what function each expert is performing, and use it for similar goals in different datasets. This modularity is a fundamental property of models which transfer well to different distributions and should be further investigated.

The modularity of a causal mechanism $Y \mid \mathbf{X}_{S^*}$ given an invariant subset S^* further supports this idea. We believe that building models which transfer to different distributions requires careful thought about which assumptions relate the different tasks and the data generating process. In particular, what are the *invariants* between the tasks which we can exploit for prediction? This thesis suggests that one possible direction is learning feature representations on top of which invariant mechanisms can be learned. While this objective is causally motivated, causality should not be seen as a bottleneck in problems in which a causal interpretation is difficult, on the contrary, exploiting invariance properties in transfer learning and reinforcement learning is a promising direction for higher level generalisation in our algorithms.

6.3 For Few-shot Learning, Learn the Best Possible Features

The results presented in Chapter 4 lead to direct advice to practitioners. The field of meta-learning has recently come to dominate the few-shot learning literature, and often involves training protocols which may be challenging for users to deploy. Our results on CIFAR-100 and *mini*ImageNet suggest that this may not strike the best trade-off in terms of complexity and accuracy. A key condition is that the few-shot classes can be differentiated by features which are also present in the training classes. In such cases, our results suggest that focusing on data augmentation and training deep, powerful features using the initial dataset achieves a higher accuracy on the few-shot classes. While something as simple as a logistic regression on top of these features performs well, the weight vectors from the training classes provide a principled way to regularise the model. Our probabilistic framework also achieves a good

trade-off in terms of accuracy and calibration, and importantly seems to strike a good balance between forgetting the old classes and learning the new ones.

Several questions remain open. First, the Gaussian model does not seem a priori to be the best fit for the weights, as the t-SNE embeddings display multi modality. Nonetheless, the small number of training classes and the high dimensionality of the feature space lead to a statistically difficult problem, where a simple model strikes a better balance. Investigating more complex models and inference procedures on datasets with more classes such as ImageNet, or datasets with large label spaces, may reveal further advantages to our framework. Second, building features when the new classes are chosen adversarially (Motiian et al., 2017) is a difficult problem, and probably requires the feature extractor to be updated using the few-shot classes.

6.4 Empirically, Features Extracted from Static Entities Preserve the Causal Footprint

The experiments in Chapter 5 show that, using proxy variables, we can construct a projected sample

$$\{(a_i, b_i)\}_{i=1}^{n}$$

which preserves the causal footprint of two static entities x and y. This is true in two settings with very different data modalities: vision and language. Further experiments could apply these methods to other examples, such as identifying the original and translated versions from a document and its translation. More importantly, the results of the language example hint at the existence of causal signals in natural language, which could in turn be exploited for reasoning. Interesting extensions include finding the causal direction when more complex statements such as “The virus spread in the city” and “Everyone in the city developed a strange set of symptoms”, are given as input to the system. These results are relevant to the ongoing $45 million dollar Big Mechanism DARPA initiative (Cohen, 2015), which aims to collect and combine causal relations from a large collection of medical documents, resulting in potentially new, undiscovered causal links to different diseases. Currently, the difficulty in applying our methods is two-fold:

- designing a proxy variable requires significant domain knowledge and can prove challenging,
- training general causal discovery algorithms able to discover a rich collection of causal footprints is still an open research question.

We believe research in both directions could lead to large-scale applications using extensions of the proxy framework.

On an orthogonal note, theoretical results on the structure of valid proxies, as well as identifiability guarantees, must be derived if proxy based methods are to be deployed with confidence in sensitive applications.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Aldrich, J. (1989). Autonomy. *Oxford Economic Papers*, pages 15 – 34.

Argyriou, A., Evgeniou, T., and Pontil, M. (2007). Multi-task feature learning. In *Proceedings of the 19th Conference on Neural Information Processing Systems (NIPS)*, pages 41 – 48.

Baayen, H. (2001). *Word Frequency Distributions*. Kluwer.

Bakker, B. and Heskes, T. (2003). Task clustering and gating for bayesian multi-task learning. *Journal of Machine Learning Research*, 4:83 – 99.

Baroni, M., Bernardi, R., Do, N., and Shan, C. (2012). Entailment above the word level in distributional semantics. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 23 – 32.

Battaglia, P., Hamrick, J., and Tenenbaum, J. (2013). Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences (PNAS)*, 110(45):18327 – 18332.

Bauer, M., Rojas-Carulla, M., Świątkowski, J., Schölkopf, B., and Turner, R. (2017). Discriminative k-shot learning using probabilistic models. *Bayesian Deep Learning Workshop of the 31st Conference on Neural Information Processing Systems (NIPS)*.

Baxter, J. (2000). A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149 – 198.

Ben-David, S., Lu, T., Luu, T., and Pál, D. (2010). Impossibility theorems for domain adaptation. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 129 – 136.

Bickel, S., Brückner, M., and Scheffer, T. (2007). Discriminative learning for differing training and test distributions. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, pages 81 – 88. ACM.

Blanco, E., Castell, N., and Moldovan, D. (2008). Causal relation extraction. In *Proceedings of the 11th Language Resources and Evaluation Conference (LREC)*.

Blumer, A., Ehrenfeucht, A., Haussler, D., and Warmuth, M. K. (1989). Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36(4):929 – 965.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics (ACL)*, 5.

Bühlmann, P. and van de Geer, S. (2011). *Statistics for High-Dimensional Data: Methods, Theory and Applications*. Springer Series in Statistics. Springer, New York, NY.

Burgess, J., Lloyd, J., and Ghahramani, Z. (2016). One-shot learning in discriminative neural networks. *Bayesian Deep Learning Workshop of the 30th Conference on Neural Information Processing Systems (NIPS)*.

Chen, M., Xu, Z., Weinberger, K., and Sha, F. (2012). Marginalized denoising autoencoders for domain adaptation. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 767 – 774.

Church, K. and Hanks, P. (1990). Word association norms, mutual information, and lexicography. *Computational linguistics*.

Clevert, D., Unterthiner, T., and Hochreiter, S. (2016). Fast and accurate deep network learning by exponential linear units (Elus). *Proceedings of 6th International Conference on Learning Representations (ICLR)*.

Cohen, P. (2015). DARPA's Big Mechanism program. *Physical biology*.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (methodological)*, pages 1 – 38.

Diuk, C., Cohen, A., and Littman, M. (2008). An object-oriented representation for efficient reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 240 – 247.

Feurer, M., Springenberg, J., and Hutter, F. (2015). Initializing bayesian hyperparameter optimization via meta-learning. In *Proceedings of the 29th Conference on Artificial Intelligence (AAAI)*, pages 1128 – 1135.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. *Proceedings of the 34th International Conference of Machine Learning (ICML)*.

French, R. (1999). Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128 – 135.

Gatys, L., Ecker, A., and Bethge, M. (2016). Image style transfer using convolutional neural networks. In *Proceedings of the 31st Conference on Computer Vision and Pattern Recognition (CVPR)*.

Geffet, M. and Dagan, I. (2005). The distributional inclusion hypotheses and lexical entailment. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 107 – 114.

Geman, S., Bienenstock, E., and Doursat, R. (1992). Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1 – 58.

Girju, R., Nakov, P., Nastase, V., Szpakowicz, S., Turney, P., and Yuret, D. (2009). Classification of semantic relations between nominals. *Language Resources and Evaluation*.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 513 – 520.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep Learning*. MIT press Cambridge.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Proceedings of the 27th Conference on Neural Information Processing Systems (NIPS)*, pages 2672 – 2680.

Gretton, A., Borgwardt, K., Rasch, M., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13:723 – 773.

Gretton, A., Bousquet, O., Smola, A., and Schölkopf, B. (2005). Measuring statistical dependence with Hilbert-Schmidt norms. In *Proceedings of the 16th Conference on Algorithmic Learning Theory (ALT)*, pages 63 – 77. Springer.

Gretton, A., Fukumizu, K., Teo, C., Song, L., Schölkopf, B., and Smola, A. (2007). A kernel statistical test of independence. In *Proceedings of the 19th Conference on Neural Information Processing Systems (NIPS)*, pages 585 – 592.

Griffiths, T. and Ghahramani, Z. (2011). The Indian buffet process: An introduction and review. *Journal of Machine Learning Research*, 12:1185 – 1224.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. (2017). On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pages 1321 – 1330.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 297 – 304.

Hammersley, J. and Handscomb, D. (1965). *Monte Carlo methods*. Monographs on statistics and applied probability. Chapman and Hall, London, New York.

Hariharan, B. and Girshick, R. (2017). Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of 16th IEEE International Conference on Computer Vision (ICCV)*.

Hashimoto, C., Torisawa, K., De Saeger, S., Oh, J., and Kazama, J. (2012). Excitatory or inhibitory: A new semantic orientation extracts contradiction and causality from the web. In *Proceedings of the Conference on Empirical Methods in Natural Language (EMNLP)*, pages 619 – 630.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the 31st IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Heinze-Deml, C., Peters, J., and Meinshausen, N. (2018). Invariant causal prediction for nonlinear models. *Journal of Causal Inference*.

Hoffman, M. and Gelman, A. (2014). The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593 – 1623.

Hoover, K. (1990). The logic of causal inference. *Economics and Philosophy*, 6:207 – 234.

Hoyer, P., Janzing, D., Mooij, J., Peters, J., and Schölkopf, B. (2009). Nonlinear causal discovery with additive noise models. In *Proceedings of the 23rd Conference on Neural Information Processing Systems (NIPS)*.

Huang, J., Smola, A., Gretton, A., Borgwardt, K., and Schölkopf, B. (2007). Correcting sample selection bias by unlabeled data. In *Proceedings of the 20th Conference on Neural Information Processing Systems (NIPS)*, pages 601 – 608, Cambridge, MA, USA. MIT Press.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37, pages 448 – 456.

Janzing, D. and Schölkopf, B. (2010). Causal inference using the algorithmic Markov condition. *IEEE Transactions on Information Theory*, 56(10):5168 – 5194.

Kansky, K., Silver, T., Mély, D., Eldawy, M., Lázaro-Gredilla, M., Lou, X., Dorfman, N., Sidor, S., Phoenix, S., and George, D. (2017). Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1809 – 1818.

Kemmeren, P., Sameith, K., van de Pasch, L., Benschop, J., Lenstra, T., Margaritis, T., O'Duibhir, E., Apweiler, E., van Wageningen, S., Ko, C., et al. (2014). Large-scale genetic perturbations reveal regulatory networks and an abundance of gene-specific repressors. *Cell*, 157(3):740 – 752.

Kilbertus, N., Rojas Carulla, M., Parascandolo, G., Hardt, M., Janzing, D., and Schölkopf, B. (2017). Avoiding discrimination through causal reasoning. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, pages 656 – 666.

Kingma, D. and Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*.

Koch, G., Zemel, R., and Salakhutdinov, R. (2015). Siamese neural networks for one-shot image recognition. *Deep Learning Workshop of the 32nd International Conference on Machine Learning (ICML)*.

Kotlerman, L., Dagan, I., Szpektor, I., and Zhitomirsky-Geffet, M. (2010). Directional distributional similarity for lexical inference. *Natural Language Engineering*, 16(4):359 – 389.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report.

Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the 26th Conference on Neural Information Processing Systems (NIPS)*, pages 1097 – 1105.

Lake, B., Salakhutdinov, R., and Tenenbaum, J. (2015). Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332 – 1338.

Lemke, C., Budka, M., and Gabrys, B. (2015). Metalearning: a survey of trends and technologies. *Artificial Intelligence Review*, 44(1):117 – 130.

Lenci, A. and Benotto, G. (2012). Identifying hypernyms in distributional semantic spaces. In *Proceedings of the 1st Conference on Lexical and Computational Semantics (*SEM)*, pages 75 – 79.

Levene, H. (1960). Robust tests for equality of variances. *Contributions to probability and statistics: Essays in honor of Harold Hotelling*, 2:278 – 292.

Little, R. and Rubin, D. (1986). *Statistical Analysis with Missing Data*. John Wiley & Sons.

Liu, D. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1):503 – 528.

Lopez-Paz, D., Muandet, K., Schölkopf, B., and Tolstikhin, I. (2015). Towards a learning theory of cause-effect inference. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*.

Lopez-Paz, D., Nishihara, R., Chintala, S., Schölkopf, B., and Bottou, L. (2017). Discovering causal signals in images. In *Proceedings of the 32nd IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Lopez-Paz, D. and Oquab, M. (2016). Revisiting classifier two-sample tests. *Proceedings of 6th International Conference on Learning Representations (ICLR)*.

Melamud, O., Levy, O., Dagan, I., and Ramat-Gan, I. (2015). A simple word embedding model for lexical substitution. In *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*.

Messerli, F. (2012). Chocolate consumption, cognitive function, and Nobel laureates. *New England Journal of Medicine*, 367:1562 – 1564.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv e-print:1301.3781*.

Mirza, P. and Tonelli, S. (2016). CATENA: CAusal and TEmporal relation extraction from NAtural language texts. In *Proceedings of the 25th Conference on Computational Linguistics (COLING)*, pages 64 – 75.

Motiian, S., Jones, Q., Iranmanesh, S., and Doretto, G. (2017). Few-shot adversarial domain adaptation. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, pages 6670 – 6680.

Muandet, K., Balduzzi, D., and Schölkopf, B. (2013). Domain generalization via invariant feature representation. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 10 – 18.

Murphy, K. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Neal, R. (2011). MCMC using Hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11).

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22:1345 – 1359.

Parascandolo, G., Kilbertus, N., Rojas-Carulla, M., and Schölkopf, B. (2018). Learning independent causal mechanisms. *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 80:4033 – 4041.

Pearl, J. (2009). *Causality: Models, Reasoning, and Inference*. Cambridge University Press, USA, 2nd edition.

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

Peters, J., Janzing, D., and Schölkopf, B. (2017). *Elements of Causal Inference: Foundations and Learning Algorithms*. MIT Press, Cambridge, MA, USA.

Peters, J., Mooij, J., Janzing, D., and Schölkopf, B. (2014). Causal discovery with continuous additive noise models. *Journal of Machine Learning Research*, 15:2009 – 2053.

Qiao, S., Liu, C., Shen, W., and Yuille, A. (2018). Few-shot image recognition by predicting parameters from activations. In *Proceedings of the 33rd IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. (2009). *Dataset shift in machine learning*. The MIT Press.

Radinsky, K., Davidovich, S., and Markovitch, S. (2012). Learning causality for news events prediction. In *World Wide Web*.

Ravi, S. and Larochelle, H. (2017). Optimization as a model for few-shot learning. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, volume 1, page 6.

Reichenbach, H. (1956). The direction of time.

Rojas-Carulla, M., Baroni, M., and Lopez-Paz, D. (2017). Causal discovery using proxy variables. *Workshop of the International Conference on Learning Representations (ICLR)*.

Rojas-Carulla, M., Schölkopf, B., Turner, R., and Peters, J. (2018). Invariant models for causal transfer learning. *Journal of Machine Learning Research*.

Roller, S., Erk, K., and Boleda, G. (2014). Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of the 22nd Conference on Computational Linguistics (COLING)*, pages 1025 – 1036.

Rosenstein, M., Marx, Z., Kaelbling, L., and Dietterich, T. (2005). To transfer or not to transfer. In *Workshop on Inductive Transfer of the 18th Conference on Neural Information Processing Systems (NIPS)*, volume 2, page 7.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A., and Fei-Fei, L. (2015). ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211 – 252.

Sahlgren, M. (2008). The distributional hypothesis. *Italian Journal of Disability Studies*, 20:33–53.

Santus, E., Lenci, A., Lu, Q., and Schulte im Walde, S. (2014). Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 38 – 42.

Schölkopf, B., Janzing, D., Peters, J., Sgouritsa, E., Zhang, K., and Mooij, J. M. (2012). On causal and anticausal learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1255 – 1262.

Schölkopf, B., Janzing, D., Peters, J., Sgouritsa, E., Zhang, K., and Mooij, J. M. (2012). On causal and anticausal learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1255 – 1262.

Schölkopf, B., Smola, A., et al. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT press.

Scott, D. (2015). *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons.

Shimodaira, H. (2000). Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference*, 90(2):227 – 244.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv e-print:1409.1556*.

Smola, A., Gretton, A., Song, L., and Schölkopf, B. (2007). A Hilbert space embedding for distributions. In *Proceedings of the 18th Conference on Algorithmic Learning Theory (ALT)*, pages 13 – 31. Springer-Verlag.

Snell, J., Swersky, K., and Zemel, R. (2017). Prototypical networks for few-shot learning. *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*.

Sriperumbudur, B. K., Gretton, A., Fukumizu, K., Schölkopf, B., and Lanckriet, G. (2010). Hilbert space embeddings and metrics on probability measures. *Journal of Machine Learning Research*, 99:1517 – 1561.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929 – 1958.

Srivastava, N. and Salakhutdinov, R. (2013). Discriminative transfer learning with tree-based priors. In *Proceedings of the 27th Conference on Neural Information Processing Systems (NIPS)*, pages 2094 – 2102.

Sugiyama, M., Nakajima, S., Kashima, H., Buenau, P., and Kawanabe, M. (2008). Direct importance estimation with model selection and its application to covariate shift adaptation. In *Proceedings of the 21st Conference on Neural Information Processing Systems (NIPS)*, pages 1433 – 1440.

Thrun, S. and Pratt, L. (1998). *Learning to learn*. Springer Science & Business Media.

Tibshirani, R. (1994). Regression selection and shrinkage via the Lasso. Technical report, Department of Statistics, University of Toronto. ftp://utstat.toronto.edu/pub/tibs/lasso.ps.

Tibshirani, R. (1996). Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society: Series B*, 58:267 – 288.

Vapnik, V. (1982). *Estimation of Dependences Based on Empirical Data*. Springer-Verlag, Berlin.

Vapnik, V. (1992). Principles of risk minimization for learning theory. In *Proceedings of the 5th Conference on Neural Information Processing Systems (NIPS)*, pages 831 – 838.

Vinyals, O., Blundell, C., Lillicrap, T., Wierstra, D., et al. (2016). Matching networks for one shot learning. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS)*, pages 3630 – 3638.

Wang, Y., Girshick, R., Hebert, M., and Hariharan, B. (2018). Low-shot learning from imaginary data. *Proceedings of the 33rd IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Weeds, J., Clarke, D., Reffin, J., Weir, D., and Keller, B. (2014). Learning to distinguish hypernyms and co-hyponyms. In *Proceedings of the 22nd Conference on Computational Linguistics (COLING)*, pages 2249 – 2259.

Weeds, J. and Weir, D. (2003). A general framework for distributional similarity. In *Proceedings of the Conference on Empirical Methods in Natural Language (EMNLP)*, pages 81 – 88.

Weeds, J., Weir, D., and McCarthy, D. (2004). Characterising measures of lexical distributional similarity. In *Proceedings of the 12th Conference on Computational Linguistics (COLING)*.

Zhang, K., Peters, J., Janzing, D., and Schölkopf, B. (2011). Kernel-based Conditional Independence Test and Application in Causal Discovery. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 804 – 813.

Zhang, K., Zhang, J., and Schölkopf, B. (2015). Distinguishing cause from effect based on exogeneity. In *Proceedings of the 15th Conference on Theoretical Aspects of Rationality and Knowledge (TARK)*.

Zwiernik, P., Uhler, C., and Richards, D. (2017). Maximum likelihood estimation for linear Gaussian covariance models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 79(4):1269 – 1292.

Appendix A

Architecture Details for the Competition of Experts

Expert
Layers
$3 \times 3, 32$, BN, ELU
$3 \times 3, 32$, BN, ELU
$3 \times 3, 32$, BN, ELU
$3 \times 3, 32$, BN, ELU
$3 \times 3, 1$, sigmoid

Discriminator
Layers
$3 \times 3, 16$, ELU
$3 \times 3, 16$, ELU
$3 \times 3, 16$, ELU
2×2, avg pooling
$3 \times 3, 32$, ELU
$3 \times 3, 32$, ELU
2×2, avg pooling
$3 \times 3, 64$, ELU
$3 \times 3, 64$, ELU
2×2, avg pooling
1024, FC, ELU
1, FC, sigmoid

Table A.1 Neural network architectures for the competition of experts.

This section presents the architecture of the experts and the discriminator for the experiments in Section 3.4.2 of Chapter 3.

The experts are fully convolutional with five convolutional layers, 32 3×3 filters per layer. We use ELU (Clevert et al., 2016) activation functions, zero padding and batch-normalisation (BN) (Ioffe and Szegedy, 2015). For further details on the architecture, see Table A.1.

The discriminator is CNN. Average pooling is applied after each two convolutional layers and the number of filters is multiplied by two. The last layer is fully connected (FC). For further details on the architecture, see Table A.1.

Both networks are optimised using Adam (Kingma and Ba, 2015) with the default hyper-parameters.

Appendix B

Datasets and Architectures for Few-shot Learning

B.1 Dataset Details for CIFAR-100

CIFAR-100 consists of 100 classes each with 500 training and 100 test images of size 32×32. The classes are grouped into 20 superclasses with 5 classes each. For example, the superclass "fish" contains the classes aquarium fish, flatfish, ray, shark, and trout. Unless otherwise stated, we used a random split into 80 base classes and 20 few-shot learning classes.

For few-shot learning and testing, we split the 100 classes into 80 base classes used for network training and 20 few-shot learning classes as follows.

```
classes_base = [
    0, 1, 2, 3, 4, 5, 6, 7, 9, 10, 13, 14, 15, 16, 17, 18, 19, 21, 22,
    24, 25, 27, 28, 32, 34, 35, 36, 38, 40, 42, 43, 44, 45, 46, 48, 49,
    50, 51, 52, 53, 54, 55, 56, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
    69, 70, 73, 74, 75, 76, 77, 78, 79, 80, 82, 83, 85, 86, 87, 88, 89,
    90, 91, 92, 93, 94, 95, 96, 97, 98, 99
]
classes_heldout = [
    8, 11, 12, 20, 23, 26, 29, 30, 31, 33, 37, 39, 41, 47, 57, 68, 71,
    72, 81, 84
]
```

B.2 Dataset Details for *mini*ImageNet

To construct *mini*ImageNet we use the same classes as initially proposed by Ravi and Larochelle (2017) and used in (Snell et al., 2017), which is split into 64 training classes (see Table B.1), 16 validation classes (see Table B.2), and 20 test classes (see Table B.3).

As we do not require a validation set, we combine the training and validation set to form an extended training set. We extract 600 images per class from the ImageNet 2012 Challenge dataset (Krizhevsky et al., 2012), scale the shorter side to 84 pixels and then centrally crop to 84×84 pixels, that is, we preserve the original aspect ratio of the image content. We use these coloured $84 \times 84 \times 3$ images as input for representational and few-shot learning and testing.

In order to train very deep models, such as a ResNet, we perform data augmentation as is the case when training full ImageNet. We use the following standard data augmentation from ImageNet that we adapt to the size of the input images:

- random horizontal flipping
- randomly paste image into 100×100 frame and cut out central 84×84 pixels
- randomly change brightness, contrast, saturation and lighting

No data augmentation is done for few-shot learning and few-shot testing.

B.3 Network Architecture and Training: ResNet Inspired

The network architecture is inspired by the ResNet-34 architecture for ImageNet (He et al., 2016) that uses convolution blocks, with two convolutions each, that are bridged by skip connections. As a base, we utilise the example code[1] provided by `tensorpack` (https://github.com/ppwwyyxx/tensorpack), a neural network training library built on top of `tensorflow` (Abadi et al., 2015). We adapt the number of features as well as the size of the last fully connected layer to account for the smaller number of training samples and training classes. For details on the architecture, see Table B.4.

The network is trained using a decaying learning rate schedule and momentum SGD and is implemented in `tensorpack` using `tensorflow`.

B.4 Network Architecture and Training: VGG Inspired

The network architecture was inspired by the VGG networks (Simonyan and Zisserman, 2014), but does not employ batch normalisation (Ioffe and Szegedy, 2015). We use exponential linear units (ELU) as activation functions. To regularise the networks, we employ

[1] https://github.com/ppwwyyxx/tensorpack/tree/master/examples/ResNet

dropout (Srivastava et al., 2014) and regularisation of the weights in the fully connected layers. The networks are trained with the ADAM optimiser (Kingma and Ba, 2015) with decaying learning rate. For details on the architecture, see Table B.5

The network is implemented in `tensorpack` using `tensorflow`.

n03400231	frying pan, frypan, skillet
n02108551	Tibetan mastiff
n02687172	aircraft carrier, carrier, flattop, attack aircraft carrier
n04296562	stage
n13133613	ear, spike, capitulum
n02165456	ladybug, ladybeetle, lady beetle, ladybird, ladybird beetle
n03337140	file, file cabinet, filing cabinet
n02966193	carousel, carrousel, merry-go-round, roundabout, whirligig
n02074367	dugong, Dugong dugon
n02105505	komondor
n04389033	tank, army tank, armored combat vehicle, armoured combat vehicle
n09246464	cliff, drop, drop-off
n03924679	photocopier
n03527444	holster
n04612504	yawl
n01749939	green mamba
n04251144	snorkel
n03347037	fire screen, fireguard
n04067472	reel
n03998194	prayer rug, prayer mat
n13054560	bolete
n02747177	ashcan, trash can, garbage can, wastebin, ash bin, ash-bin, ashbin, dustbin, trash barrel, trash bin
n04435653	tile roof
n02108089	boxer
n03908618	pencil box, pencil case
n01770081	harvestman, daddy longlegs, Phalangium opilio
n03676483	lipstick, lip rouge
n03220513	dome
n04515003	upright, upright piano
n04258138	solar dish, solar collector, solar furnace
n04509417	unicycle, monocycle
n01704323	triceratops
n04443257	tobacco shop, tobacconist shop, tobacconist
n02089867	Walker hound, Walker foxhound
n01910747	jellyfish
n02111277	Newfoundland, Newfoundland dog
n04243546	slot, one-armed bandit
n01558993	robin, American robin, Turdus migratorius
n03047690	clog, geta, patten, sabot
n03854065	organ, pipe organ
n03476684	hair slide
n02113712	miniature poodle
n07747607	orange
n03838899	oboe, hautboy, hautbois
n07584110	consomme
n02795169	barrel, cask
n03017168	chime, bell, gong
n04275548	spider web, spider's web
n04604644	worm fence, snake fence, snake-rail fence, Virginia fence
n02606052	rock beauty, Holocanthus tricolor
n01843383	toucan
n02457408	three-toed sloth, ai, Bradypus tridactylus
n03062245	cocktail shaker
n03207743	dishrag, dishcloth
n02108915	French bulldog
n06794110	street sign
n02823428	beer bottle
n03888605	parallel bars, bars
n04596742	wok
n02091831	Saluki, gazelle hound
n02101006	Gordon setter
n02120079	Arctic fox, white fox, Alopex lagopus
n01532829	house finch, linnet, Carpodacus mexicanus
n07697537	hotdog, hot dog, red hot

Table B.1 Training classes for *mini*ImageNet as proposed by Ravi and Larochelle (2017)

n03075370	combination lock
n02971356	carton
n03980874	poncho
n02114548	white wolf, Arctic wolf, Canis lupus tundrarum
n03535780	horizontal bar, high bar
n03584254	iPod
n02981792	catamaran
n03417042	garbage truck, dustcart
n03770439	miniskirt, mini
n02091244	Ibizan hound, Ibizan Podenco
n02174001	rhinoceros beetle
n09256479	coral reef
n02950826	cannon
n01855672	goose
n02138441	meerkat, mierkat
n03773504	missiles

Table B.2 Validation classes for *mini*ImageNet as proposed by Ravi and Larochelle (2017)

n02116738	African hunting dog, hyena dog, Cape hunting dog, Lycaon pictus
n02110063	malamute, malemute, Alaskan malamute
n02443484	black-footed ferret, ferret, Mustela nigripes
n03146219	cuirass
n03775546	mixing bowl
n03544143	hourglass
n04149813	scoreboard
n03127925	crate
n04418357	theater curtain, theatre curtain
n02099601	golden retriever
n02219486	ant, emmet, pismire
n03272010	electric guitar
n04146614	school bus
n02129165	lion, king of beasts, Panthera leo
n04522168	vase
n07613480	trifle
n02871525	bookshop, bookstore, bookstall
n01981276	king crab, Alaska crab, Alaskan king crab, Alaska king crab, Paralithodes camtschatica
n02110341	dalmatian, coach dog, carriage dog
n01930112	nematode, nematode worm, roundworm

Table B.3 Test classes for *mini*ImageNet as proposed by Ravi and Larochelle (2017)

ResNet-34 inspired for *mini*ImageNet	
Output size	**Layers**
$84 \times 84 \times 3$	Input patch
$42 \times 42 \times 32$	$5 \times 5, 32$, stride 2
$42 \times 42 \times 32$	$\begin{bmatrix} 3 \times 3, 32 \\ 3 \times 3, 32 \end{bmatrix} \times 3$
$21 \times 21 \times 64$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 4$
$11 \times 11 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 6$
$6 \times 6 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 3$
256	global average pooling
$\widetilde{C}$	fully connected, softmax

Table B.4 Network architecture. All unnamed layers are 2D convolutions with stated kernel size and padding SAME; the output of the shaded layer corresponds to $\Phi_\varphi(\mathbf{u})$, the feature space representation of the image $\mathbf{u}$, which is used as input for probabilistic few-shot learning.

VGG-style Network for CIFAR-100	
Output size	**Layers**
$32 \times 32 \times 3$	Input patch
$16 \times 16 \times 64$	2× (Conv2D, ELU), Pool
$8 \times 8 \times 64$	2× (Conv2D, ELU), Pool
$4 \times 4 \times 128$	2× (Conv2D, ELU), Pool
$2 \times 2 \times 128$	2× (Conv2D, ELU), Pool
$2 \times 2 \times 128$	Dropout (0.5)
256	FullyConnected, ELU
256	Dropout (0.5)
128	FullyConnected, ELU
$\widetilde{C}$	FullyConnected, SoftMax

VGG-style Network for *mini*ImageNet	
Output size	**Layers**
$84 \times 84 \times 3$	Input patch
$42 \times 42 \times 32$	2× (Conv2D, ELU), Pool
$21 \times 21 \times 64$	2× (Conv2D, ELU), Pool
$11 \times 11 \times 128$	2× (Conv2D, ELU), Pool
$6 \times 6 \times 128$	2× (Conv2D, ELU), Pool
$3 \times 3 \times 128$	2× (Conv2D, ELU), Pool
$3 \times 3 \times 128$	Dropout (0.5)
512	FullyConnected, ELU
512	Dropout (0.5)
256	FullyConnected, ELU
$\widetilde{C}$	FullyConnected, SoftMax

Table B.5 Network architectures. All 2D convolutions have kernel size 3×3 and padding SAME; max-pooling is performed with stride 2. The output of the shaded layer corresponds to $\Phi_\varphi(u)$, the feature space representation of the image u, which is used as input for probabilistic few-shot learning.

Appendix C

Instructions for Creating the Dataset of Word Pairs

C.1 Instructions for Word Pair Creators

We will ask you to write word pairs (for instance, WordA and WordB) for which you believe the statement "WordA causes WordB" is true.

To provide us with high quality word pairs, we ask you to follow these indications:

- All word pairs must have the form "WordA → WordB". It is essential that the first word (WordA) is the cause, and the second word (WordB) is the effect.
- WordA and WordB must be one word each (no spaces, and no "recessive gene → red hair"). Avoid compound words such as "snow-blind".
- In most situations, you may come up with a word pair that can be justified both as "WordA → WordB" and "WordB → WordA". In such situations, prefer the causal direction with the easiest explanation. For example, consider the word pair "virus → death". Most people would agree that "virus causes death". However, "death causes virus" can be true in some specific scenario (for example, "because of all the deaths in the region, a new family of virus emerged."). However, the explanation "virus causes death" is preferred, because it is more general and depends less on the context.
- We do not accept word pairs with an ambiguous causal relation, such as "book - paper".
- We do not accept simple variations of word pairs. For example, if you wrote down "dog → bark", we will not credit you for other pairs such as "dogs → bark" or "dog → barking".
- Use frequent words (avoid strange words such as "clithridiate").

- Do not rely on our examples, and use your creativity. We are grateful if you come up with diverse word pairs! Please do not add any numbers (for example, "1 - dog → bark"). For your guidance, we provide you examples of word pairs that belong to different categories. Please bear in mind that we will reward your creativity: therefore, focus on providing new word pairs with an evident causal direction, and do not limit yourself to the categories shown below.

1) Physical phenomenon: there exists a clear physical mechanism that explains why "WordA → WordB".

- sun → radiation (The sun is a source of radiation. If the sun were not present, then there would be no radiation.)
- altitude → temperature
- winter → cold
- oil → energy

2) Events and consequences: WordA is an action or event, and WordB is a consequence of that action or event.

- crime → punishment
- accident → death
- smoking → cancer
- suicide → death
- call → ring

3) Creator and producer: WordA is a creator or producer, WordB is the creation of the producer.

- writer → book (the creator is a person)
- painter → painting
- father → son
- dog → bark
- bacteria → sickness
- pen → drawing (the creator is an object)

- chef → food
- instrument → music
- bomb → destruction
- virus → death

4) Other categories! Up to you, please use your creativity!

- fear → scream
- age → salary

C.2 Instructions for Word Pair Validators

Please classify the relation between pairs of words A and B into one of three categories: either "A causes B", "B causes A", or "Non-causal or unrelated".

For example, given the pair of words "virus and death", the correct answer would be:

- virus causes death (correct);
- death causes virus (wrong);
- non-causal or unrelated (wrong).

Some of the pairs that will be presented are non-causal. This may happen if:

- The words are unrelated, like "toilet and beach".
- The words are related, but there is no clear causal direction. This is the case of "salad and lettuce", since we can eat salad without lettuce, or eat lettuce in a burger.

To provide us with high quality categorization of word pairs, we ask you to follow these indications:

- Prefer the causal direction with the simplest explanation. Most people would agree that "virus causes death". However, "death causes virus" can be true in some specific scenario (for example, "because of all the deaths in the region, a new virus emerged."). However, the explanation "virus causes death" is preferred, because it is true in more general contexts.
- If no direction is clearer, mark the pair as non-causal. Here, conservative is good!
- Think twice before deciding. We will present the pairs in random order!

Please classify all the presented pairs. If one or more has not been answered, the whole batch will be invalid. **PLEASE DOUBLE CHECK THAT YOU HAVE ANSWERED ALL 40 WORD PAIRS.**

Examples of causal word pairs:

- "sun and radiation": sun causes radiation
- "energy and oil": oil causes energy
- "punishment and crime": crime causes punishment
- "instrument and music": instrument causes music
- "age and salary": age causes salary

Examples of non-causal word pairs:

- "video and games": non-causal or unrelated
- "husband and wife": non-causal or unrelated
- "salmon and shampoo": non-causal or unrelated
- "knife and gun": non-causal or unrelated
- "sport and soccer": non-causal or unrelated