Université de Montréal

AI Alignment and Generalization in Deep Learning

par
David Krueger

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en computer science

décembre, 2021

Université de Montréal
Faculté des études supérieures

Cette thèse intitulée:

AI Alignment and Generalization in Deep Learning

présentée par:

David Krueger

a été évaluée par un jury composé des personnes suivantes:

Guillaume Lajoie,	président-rapporteur
Aaron Courville,	directeur de recherche
Yoshua Bengio,	membre du jury
Roger Grosse,	examinateur externe
Jonathan Simon,	représentant du doyen de la FES

Thèse acceptée le: . . . . . .

RÉSUMÉ

Cette thèse discute quelques travaux en apprentissage profond visant à comprendre et à améliorer les capacités de généralisation des réseaux de neurones profonds (DNN). Les DNNs atteignent des performances inégalées dans un éventail croissant de tâches et de domaines, mais leur comportement pendant l'apprentissage et le déploiement reste mal compris. Ils peuvent également être étonnamment fragiles : la généralisation dans la distribution peut être un mauvais prédicteur du comportement ou de la performance lors de changements de distribution, ce qui ne peut généralement pas être évité dans la pratique. Bien que ces limitations ne soient pas propres aux DNN - et sont en effet susceptibles de constituer des défis pour tout système d'IA suffisamment complexe - la prévalence et la puissance des DNN les rendent particulièrement dignes d'étude.

J'encadre ces défis dans le contexte plus large de «l'alignement de l'IA» : un domaine naissant axé sur la garantie que les systèmes d'IA se comportent conformément aux intentions de leurs utilisateurs. Bien que rendre les systèmes d'IA plus intelligents ou capables puisse aider à les rendre plus alignés, cela n'est ni nécessaire ni suffisant pour l'alignement. Cependant, être capable d'aligner les systèmes d'IA de pointe (par exemple les DNN) est d'une grande importance sociale afin d'éviter les comportements indésirables et dangereux des systèmes d'IA avancés. Sans progrès dans l'alignement de l'IA, les systèmes d'IA avancés pourraient poursuivre des objectifs contraires à la survie humaine, posant un risque existentiel («x-risque») pour l'humanité.

L'un des principes fondamentaux de cette thèse est que l'obtention de hautes performances sur les repères d'apprentissage automatique est souvent un bon indicateur des capacités des systèmes d'IA, mais pas de leur alignement. En effet, les systèmes d'IA atteignent souvent des performances élevées de manière inattendue, ce qui révèle les limites de nos mesures de performance et, plus généralement, de nos techniques pour spécifier nos intentions. L'apprentissage des intentions humaines à l'aide des DNN est quelque peu prometteur, mais les DNN sont toujours enclins à apprendre à résoudre des tâches en utilisant des concepts de «caractéristiques» très différents de ceux qui sont saillants pour les humains. En effet, c'est une source majeure de leur mauvaise généralisation sur les

données hors distribution.

En comprenant mieux les succès et les échecs de la généralisation DNN et les méthodes actuelles de spécification de nos intentions, nous visons à progresser vers des systèmes d'IA basés sur l'apprentissage en profondeur qui sont capables de comprendre les intentions des utilisateurs et d'agir en conséquence.

mots clés: Sécurité de l'IA, apprentissage automatique, Réseaux neuronaux, Généralisation hors de la distribution, Généralisation du domaine, Apprentissage profond Bayésien, Flux de normalisation, Prédiction invariante, Risque existentiel.

ABSTRACT

This thesis covers a number of works in deep learning aimed at understanding and improving generalization abilities of deep neural networks (DNNs). DNNs achieve unrivaled performance in a growing range of tasks and domains, yet their behavior during learning and deployment remains poorly understood. They can also be surprisingly brittle: in-distribution generalization can be a poor predictor of behavior or performance under distributional shifts, which typically cannot be avoided in practice. While these limitations are not unique to DNNs – and indeed are likely to be challenges facing any AI systems of sufficient complexity – the prevalence and power of DNNs makes them particularly worthy of study.

I frame these challenges within the broader context of "AI Alignment": a nascent field focused on ensuring that AI systems behave in accordance with their user's intentions. While making AI systems more intelligent or capable can help make them more aligned, it is neither necessary nor sufficient for alignment. However, being able to align state-of-the-art AI systems (e.g. DNNs) is of great social importance in order to avoid undesirable and unsafe behavior from advanced AI systems. Without progress in AI Alignment, advanced AI systems might pursue objectives at odds with human survival, posing an existential risk ("x-risk") to humanity.

A core tenet of this thesis is that the achieving high performance on machine learning benchmarks if often a good indicator of AI systems' capabilities, but not their alignment. This is because AI systems often achieve high performance in unexpected ways that reveal the limitations of our performance metrics, and more generally, our techniques for specifying our intentions. Learning about human intentions using DNNs shows some promise, but DNNs are still prone to learning to solve tasks using concepts of "features" very different from those which are salient to humans. Indeed, this is a major source of their poor generalization on out-of-distribution data.

By better understanding the successes and failures of DNN generalization and current methods of specifying our intentions, we aim to make progress towards deep-learning based AI systems that are able to understand users' intentions and act accordingly.

Keywords: artificial intelligence, machine learning, AI safety, neural networks, out-of-distribution generalization, domain generalization, Bayesian deep learning, normalizing flows, invariant prediction, existential risk.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

AAF	Affine Autoregressive Flow
ADA	Adversarial Domain Adaptation
AI	Artificial Intelligence
AUC	Area Under Curve
BbB	Bayes-by-Backprop
BHN	Bayesian Hypernetwork
C-ADA	Conditional Adversarial Domain Adaptation
CGM	Causal Graphical Model
CIFAR	Canadian Institute For Advanced Research
CMNIST	Colored MNIST
CNN	Convolutional Neural Network
CSR	Critical Sample Ratio
DDGN	Deep Directed Generative Network
DDSF	Deep Dense Sigmoidal Flow
DL	Deep Learning
DNN	Deep Neural Network
DRO	Distributionally Robust Optimization
DSF	Deep Sigmoidal Flow
ELBO	Evidence Lower Bound
ERM	Empirical Risk Minimization
FGS	Fast Gradient Sign
FGSM	Fast Gradient Sign Method
FMA	Fixed Mechanism Assumption
FRA	Fixed Relationship Assumption
GAN	Generative Adversarial Network
HCH	Humans Consulting HCH (N.B. this is a recursive acronym)
i.i.d.	independently and identically distributed
IAF	Inverse Autoregressive Flow
ICP	Invariant Causal Prediction
IRM	Invariant Risk Minimization
LASS	Langevin Adversarial Sample Search
MADE	Masked Autoregressive Density Estimator
MAE	Mean Absolute Error
MAF	Masked Autoregressive Flow
MAP	Maximum a Posteriori
MC	Dropout Monte Carlo Dropout
MCMC	Markov Chain Monte Carlo
ML	Machine Learning

MLE	Maximum Likelihood Estimation
MLP	Multilayer Perceptron
MM-REx	minimax Risk Extrapolation
MNIST	Modified National Institute of Standards and Technology database
MSE	Mean Squared Error
NAF	Neural Autoregressive Flow
NF	Normalizing Flow
NLL	Negative Log-likelihood
OOD	Out-of-Distribution
PACS	Photo, Art Painting, Cartoon, and Sketch datasets
PR	Precision Recall
RealNVP/RNVP	Real Non-Volume Preserving
ReLU	Recitifed Linear Unit
REx	Risk Extrapolation
RI	Risk Interpolation
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RO	Robust Optimization
ROC	Receiver Operator Characteristic
SCM	Structural Causal Model
SEM	Structural Equation Model
SGD	Stochastic Gradient Descent
V-REx	Variance Risk Extrapolation
VAE	Variational Autoencoder
VLCS	PASCAL VOC 2007, LabelMe, Caltech and Sun datasets
x-risk	existential risk
x-safety	existential safety

CHAPTER 1

INTRODUCTION

This thesis presents several works relating to the topics of 1) generalization in deep learning and 2) Bayesian Deep Learning through an organizing lens of Artificial Intelligence (AI) Alignment. It is a thesis by articles, including the following 4 publications:

1. A Closer Look at Memorization in Deep Networks [9]
2. Out-of-Distribution Generalization via Risk Extrapolation (REx) [177]
3. Bayesian Hypernetworks [176]
4. Neural Autoregressive Flows [149]

I am a joint first author on all of these publications, except for (2), for which I am the sole first author. These articles form the bulk of this thesis. I provide technical background and some historical context for each of the articles in the corresponding chapters.

In this introduction, I first describe relevant technical concepts from the fields of machine learning, deep learning, and reinforcement learning (Section 1.1). Next, I outline arguments that advances in artificial intelligence are liable to increase **existential risk (x-risk)**, i.e. lead to the extinction of human civilization (Section 1.2).[1] I then discuss approaches to reducing AI x-risk – or phrased positively, to *increasing* **x-safety**; here I focus particularly on the technical subject of **AI Alignment**, and highlight both its promise and its limitations (Section 1.3). I also (briefly) cover some of my other PhD work to provide additional context.

After covering this high-level background, I discuss the motivation for my works on generalization and Bayesian Deep Learning from the point of view of AI Alignment.The goal here is to explain the ways in which I think my work could increase x-safety by fitting it into a broader description of my motivations and perspective. I want to emphasize,

[1] Note that x-risk has been defined in subtly different ways by Bostrom [32] (originally) vs. Critch and Krueger [63] vs. Ord [226]

however, that my main reason for choosing these four projects was not because I thought they would increase x-safety. Nonetheless, in retrospect, I believe all likely make *some* progress towards the technical goal of AI Alignment. Still, it is hard to say when and whether technical progress on AI Alignment actually increases x-safety – an issue we discuss in greater length in Section 1.2.1.3.

1.1 Artificial Intelligence

Here I briefly describe the fields of Artificial Intelligence (AI), and Machine Learning (ML), Deep Learning, and Reinforcement Learning. See my Master's Thesis for a more in-depth discussion [175].

Artificial Intelligence refers to efforts to understand and engineer intelligent software systems. The computing medium (e.g. carbon- vs. silicon-based) is not important, except inasmuch as it raises or addresses practical engineering challenges and/or fundamental physical limits. The focus is on the underlying computational principles and methods by which intelligent behavior can be manifested.

Intelligence can be defined in many different ways. Legg and Hutter [188] collect 70 definitions, and noting commonalities, summarize them as: "Intelligence measures an agent's ability to achieve goals in a wide range of environments." Still, AI is a somewhat vague term; what counts as intelligent behavior is controversial, and historically, researchers have often been surprised by how much "intelligence" is involved in various human capabilities that do not obviously seem to exemplify the peak of human intelligence; examples include perception and locomotion.

Pioneers of artificial intelligence were often concerned with replicating human intelligence – as well as the consequences of doing so. Notably, Alan Turing and Norbert Wiener both considered the possibility that humanity would lose control over AI systems [63, 287, 307].

As the field has developed, many researchers have chosen to focus on "narrow AI": seeking to replicate more specific aspects or feats of human intelligence. This is sometimes distinguished from Artificial General Intelligence (AGI), which aims at

producing AI systems that have the same breadth of intelligence as humans. Roughly speaking, AI can be considered as composed of many different capabilities. Reaching or surpassing human-level performance in one narrow capability does not necessarily constitute or reflect significant progress towards AGI. Indeed, computer systems have long outperformed humans at some narrow forms of intelligence (such as quickly adding large numbers), while lagging far behind in others (such as driving cars).

Whether recent progress in narrow AI should be viewed as significant progress towards AGI is a subject of much debate. On the one hand, there are many examples where progress in one area did not transfer to others, and there are so many human capabilities that seem to require intelligence that replicating any particular one would seem like a mere drop in the bucket. On the other hand, recent progress in AI has largely been driven by **Deep Learning**, a set of highly general machine learning algorithms, which appear capable of replicating – at least to some extent – many diverse feats of human intelligence, from perception [174] and motor control [191], to mathematical reasoning [202, 235] and strategic thinking [262]. We discuss Deep Learning more in section 1.1.2.

We can also distinguish between the generality of a learning algorithm, and the generality of the knowledge or capabilities that it produces. It might be possible to use the same learning algorithm to build AI systems capable of replicating many feats of human intelligence. But the resulting AI systems might themselves still be narrow.

Deep Learning algorithms are not only highly general, they also produce AI systems of impressive (although still quite limited) generality. So called "Foundation Models" [30] – large Deep Learning models trained on large, diverse datasets and exhibiting correspondingly broad capabilities – have become a major component of state-of-the-art systems across a wide range of tasks in both computer vision [240] and natural language processing.

1.1.1 Machine Learning

It is now widely acknowledged that programming intelligent behavior directly can be prohibitively challenging. Instead, modern approaches to AI typically focus on **machine**

learning (ML) algorithms, which allow computers to learn intelligent behavior from data. Machine Learning programs perform a data-driven search for behaviors that are deemed desirable (e.g. intelligent) according to some **evaluation procedure**.

But even in the machine learning paradigm, programmers still need to specify the behavior of the AI system indirectly; they do this by defining how it will learn. This can be decomposed into four choices:

1. **Data:** What data will the AI system learn from? Will it be a fixed dataset? Will the AI play a role in collecting the data? How will human labor be used to curate and annotate the data?

2. **Training signal:** What drives and directs the learning process? How are behaviors evaluated, discouraged/encouraged, identified, and/or selected? The standard approach is to provide a real-valued score (as in a game), and seek behaviors that increase this score. This score is often called one of the following: objective, reward, cost, loss, risk, fitness, or performance. Often the programmer will specify a mathematical function mapping behaviors to scores, e.g. a loss *function*.

3. **Model:** Which kinds of behaviors do we allow to be learned? A model typically specifies a flexible family of behaviors which are modulated by a set of tuneable **parameters**, most commonly denoted θ.

4. **Learning Algorithm:** How do we search for high-scoring behaviors?

As a concrete example, we'll consider LeNet [186], a classic machine learning model that learned to classify hand-written digits using the well-known MNIST dataset, which the authors also introduced. This is a **supervised learning** task, where the goal is to predict a **target** or **label** (the digit class) from an **input** (the image of the digit). Supervised learning is probably the most popular type of machine learning task. Even when labels are not available – this is called ***un*supervised learning** – it is common to invent a supervised learning problem, by predicting part of the data from the rest of the data – in modern deep learning this is often called **self-supervised learning**. A common example is "next-step

prediction", where we try to predict the present time-step from previous time-steps for inputs that consist of a sequence of temporal observations.

We'll now describe each of the four components of the LeNet Machine Learning system:

1. **Data:** LeNet was trained **offline**, i.e. on a dataset that had already been collected and curated. The dataset was created by post-processing an existing collection of hand-written digits written by different people. The processing included centering and resizing the digits.

2. **Training Signal:** For every input image, LeNet outputs a probability distribution over all the possible labels (0 through 9). It was trained to minimize the negative log-likelihood (NLL) of the observed label under this probability distribution, averaged over all the (image, label) pairs. This loss function heavily penalizes confident, incorrect predictions, and is only minimized when every example used to train the model is predicted with absolute confidence.

3. **Model:** LeNet is an example of a Convolutional Neural Network (CNN) [185]. This is a model inspired by the visual system of animals and includes a hierarchy of learned "feature detectors". It also applies the same feature detector to different image patches, and this leads to (approximate) translation invariance, encoding the prior knowledge that the identity of a digit doesn't depend on its location within the image. The parameters of the model determine which patterns the feature detectors respond to, and also how the presence and strength of those patterns should be weighed in determining the probability distribution over labels.

4. **Learning Algorithm:** LeNet used a variant of Stochastic Gradient Descent (SGD) for learning. This algorithm repeatedly estimates the direction of steepest descent in loss (considered as a function of the parameters) and then adjusts the parameters in that direction.

These choices – training a CNN offline on the NLL loss function using SGD – reflect standard practice in Machine Learning for categorizing images according to how they are

labeled, a task called **(image) classification**.

1.1.1.1 Evaluation in Machine Learning

The training signal is often closely related to – and conflated with – the evaluation procedure. For instance, it is common to train and evaluate an ML algorithm using the same loss function (e.g. the **negative log-likelihood (NLL)** of the data under a learned statistical model), but using different data for training and evaluation (also called "testing"). However, it is also common for these to differ somewhat; for instance, likelihood may be used as a training signal, but accuracy (that is, what percentage of predictions were correct) may be used for evaluation. While researchers often focus on simple evaluation metrics as a measure of algorithmic progress, responsible deployment of AI systems may require extensive real-world testing and qualitative human judgments.

The goal of learning is often framed as 'learning a good setting of the parameters'. However, the behavior of a model depends not only on the parameter values, but also on the context in which it operates. ML algorithms can influence the world in a way that makes evaluation challenging. For instance, predictive policing models have been criticized for creating a self-fulfilling prophecy: sending more police to an area increases arrests, making it seem as if that area has more crime. These models may accurately predict arrests *only* because of the influence they have on determining policing practices.

i.i.d. data It is very common to train and evaluate ML algorithms on disjoint sets of data taken from the same underlying distribution. This setting – where all of the data the algorithm encounters is **independently and identically distributed (i.i.d.**) – can greatly simplify experimental and theoretical analysis of algorithms, but also fails to address important challenges with deploying ML in the real world. In practice, the distribution of data a system encounters in deployment is typically not i.i.d. Consider news content: the distribution of words is constantly changing as new people and topics enter the news.

Still, even in the i.i.d. setting, there can be a significant difference between performance on the data used for training (the **training set**) and that used for evaluation (the **test set**).Intuitively, we might expect an ML algorithm to perform better on data which it

has been exposed to, all else being equal; the i.i.d. learning setting effectively enforces that all else is indeed equal – a dataset is typically randomly partitioned into training and test sets. Whereas a lookup table can perform well on the observed training data, the ability to perform well on unseen test data is the "magic" of ML, and is referred to as **generalization**.

Generalization is a key desiderata for ML algorithms. A system which fails to generalize is said to have **overfit** the training data. Overfitting has traditionally been one of the core challenges in ML, however, modern deep learning methods have proven to generalize remarkably well, a phenomenon which is still not well understood. Understanding how DNNs generalize is the subject of the work presented in Section 1.4.1.

out-of-distribution generalization While i.i.d. generalization might sometimes be suitable as an indicator of various forms of narrow intelligence, it is not suitable as a measure of *general* intelligence as defined by Legg and Hutter [188], since it only considers performance within a single "environment". A system deployed in the real world may regularly encounter new environments, where background conditions are different, and the distribution of new data points does not match the distribution of previously collected data. Such data is sometimes called **out-of-distribution (OOD)**. Generalizing to OOD data is more challenging than i.i.d. generalization; even determining whether some new data is OOD is a challenging, unsolved, problem. [2]

In order to generalize OOD, it may be necessary for an AI system to have a better model of how the world actually works. Such an understanding helps to anticipate and account for the effects of changes in the background conditions on the distribution of data. A model of the world can be thought of as composed of concepts and the (logical, physical, causal, etc.) relations between them. While classic approaches to AI often represented such world models in a purely symbolic form, such an approach is fundamentally limited by the **symbol grounding problem**: intelligent behavior requires understanding how the concepts in a symbolic models are expressed in terms of sensory perceptions and actions.

[2] Note that the distinction between i.i.d. and OOD is not as crisp as we've made it out to be. In particular, since different data distributions can have overlapping support (i.e. they can both assign non-zero probability to the same example), which distribution a given example was sampled from is not necessarily well defined.

The success of Deep Learning as an approach to perception and of Deep Reinforcement Learning as an approach to action selection make these techniques promising approaches to the symbol grounding problem.

1.1.2 Deep Learning

In the last decade, Deep Learning has gone from a niche subject to the most popular area of artificial intelligence research. Deep Learning is clearly incredibly useful across a wide range of Machine Learning tasks, and it has been a core part of most of the recent high-profile advances and accomplishments in AI.

Besides its empirical success, there are also strong conceptual reasons why Deep Learning is a good approach to symbol grounding. Deep Learning methods learn (deep) **representations** of data which may be closer to human concepts than the default encoding of the data; this is thought to be a key property underlying their success. For instance, images are typically encoded as a set of pixel intensity values along 3 color "channels" (e.g. Red, Green, and Blue (RGB)), with the intensity indicating how much light of that color should be output at that pixel in order to generate the image on a computer monitor. However, when a human being looks at an image on a computer screen, they do not perceive it as a grid of pixels; they perceive it as (e.g.) a visual scene which includes objects with different shapes and textures inhabiting various locations. Deep Learning seems to find representations that capture similar abstractions, effectively grounding visual concepts in terms of low-level input features. However, the correspondence with human concepts is clearly imperfect, and this is the subject of much research.

Technically speaking, Deep Learning refers to machine learning methods that learn a composition of functions: $f \doteq f^L \circ f^{L-1} \circ \ldots f^1$. In practice, however, Deep Learning is often used as a synonym for deep neural networks (DNNs), currently the most popular type of deep learning model.

Neural networks are a broad family of machine learning models originally inspired by neuroscientific theories of how neurons work in biological organisms. Neural networks are composed of computational **units**, frequently referred to as **neurons**, connected by weighted edges, called **weights**. The neurons compute values, called **activations** based

on the values of the neurons they are connected with, and the weights of the edges connecting them.

Edges are typically directed, with the set of units and edges form a Directed Acyclic Graph (DAG), and the activation for a given unit, h computed as a weighted sum of it's inputs, x multiplied by the edges' weights, w:

$$h(x) = \sigma\left(\sum_i x_i, w_i + b\right) \tag{1.1}$$

with an additional constant bias term, b, and nonlinearity σ.

The nonlinearity is crucial, as it increases the **capacity**, i.e. the range of functions that a DNN can represent – composing affine functions yields another affine function, but the nonlinearities in DNNs allow deeper networks (i.e. those with a deeper DAG) to express more complicated functions. The prototypical activation function of modern deep learning is the Rectified Linear Unit (ReLU), $\sigma(z) \doteq \max\{0, z\}$.

The classic form of deep neural network is the Multilayer Perceptron (MLP). An MLP is organized into a sequence or stack of **layers** of multiple units, each connected to the units of the previous layer. At the bottom of the stack are units that have no incoming edges, called **input units**, and at the top are units with no outgoing edges, **output units**. For a given (vector-valued) input $\mathbf{x}$, the activations of a given layer can be computed as $f^l \circ f^{l-1} \circ \ldots f^1(\mathbf{x})$.

Just as biological learning (often) occurs as a result of strengthening or weakening synapses between neurons, DNNs learn by adjusting the weights connecting their neurons; the weights and biases of a DNN are the parameters of the model.

1.1.3 Reinforcement Learning

Here we will begin by describing Reinforcement Learning (RL), another popular technique in modern AI, before discussing the theory of rational agency implicit in this approach. A canonical reference for Reinforcement Learning is Sutton and Barto [274] Methods combining Deep Learning and RL (creatively referred to as "Deep RL") are

behind many of the successes of super-human game-playing AI systems, including for backgammon [281], ATARI video games [214], Go and other board games [261, 261], Dota 2 [24], and StarCraft 2 [299]. And they are an active and promising area of research in robotics (although more classical methods are still more popular in practice) and automated reasoning (such as theorem proving and code generation). Deep RL is also commonly used in large-scale content recommendation algorithms that power platforms such as Facebook [218] and Youtube [49].

Like Deep Learning, Reinforcement Learning refers to a set of machine learning techniques. However, RL *also* refers to a set of **problem statements**. A problem statement is a formalization of a type of learning task, used to design and analyze learning algorithms. Like all formalisms, a problem statement may fail to capture important aspects of reality.

The RL problem statement formalizes a notion of goal-directed behavior with imperfect information. In RL, the AI system and its learning algorithm are referred to as an **agent**. The RL agent interacts with some **environment**, taking **actions** that influence the **state** of the environment, while collecting **reward**. The behavior of an RL agent in any given situation is determined by its **policy**. The goal of an RL algorithm or agent is to maximize reward. This might mean learning a (fixed) policy which is able to collect high reward when deployed. Or it could mean maximizing the total reward collected over the course of the agent's existence, while continuously learning and refining its policy. Future rewards are typically discounted; this can be thought of as analogous to monetary inflation, and is a straightforward way of avoiding technical/philosophical issues raised by the possibility of collecting infinite reward [10, 36, 306].

Consider the example of a chess playing agent, for which actions could be chess moves, the state could be the state of the board (maybe with some history of past moves), and the reward could be 1 for winning, -1 for losing and 0 for a draw. Rewards could also be given for capturing the opponents pieces, or other signs of progress, and this might make the learning problem easier, by providing more and faster feedback to the agent; this approach is known as **reward shaping**. There are methods of reward shaping that do not change the ranking of policies [221], but in general, reward shaping can lead to

suboptimal performance. For instance, in chess rewarding an agent for capturing pieces might lead it to forgo a chance to checkmate the opponent. The philosophy of RL is that the agent should discover which sub-goals, or **instrumental goals** are (most) useful on its own, and keep its "eye on the prize" of the "true" reward function. Of course, this (tacitly) assumes that there *is* a true reward function that the designer has provided, and that captures their actual preferences about the agents behavior. AI Alignment can be motivated by the observation that, in practice, we often have no idea how to provide such a reward function or otherwise communicate our preferences about the behavior of an AI system.

While there are many different problem statements in RL, I will describe what I consider the most prototypical setting for concreteness. Here, the environment is a Markov Decision Process (MDP), defined by:

- a finite set of states, $\mathscr{S}$,
- an initial state distribution $P(S_0)$
- a finite set of actions, $\mathscr{A}$,
- a **transition function**, $\mathscr{T} : \mathscr{S} \times \mathscr{A} \to \mathscr{P}(\mathscr{S})$,
- a **reward function**, $\mathscr{R} : \mathscr{S} \to \mathbb{R}$,
- a **discount factor**, γ

The agent executes a policy, $\pi_t : \mathscr{S} \to \mathscr{P}(\mathscr{A})$. Interaction between the agent and environment is defined as follows:

- At each (discrete) time-step, t, the agent observes the current state, s_t, and reward r_t.
- The agent may update its policy (e.g. to account for these new observations), using some learning algorithm, producing π_{t+1}.
- The agent samples an action $a_t \sim \pi_{t+1}(s_t)$.

- Then the environment responds by sampling a new state $s_{t+1} \sim T(s_t, a_t)$ and reward, r_{t+1}.

The agent's performance is defined as the discounted sum of rewards,

$$\sum_{t=0}^{\infty} \gamma^t R_t. \tag{1.2}$$

1.1.4 On the Implicit Normative Assumptions of Rational Agency

Like deep learning, RL is heavily inspired by theories of natural intelligence, specifically, operant conditioning. Notably, RL uses the notion of rational agency as a normative standard. In essence, a rational agent is simply one that seeks to maximize some expected utility function. Rational agency is a popular model for explaining intelligent behavior across a number of fields, e.g. in social science. Besides AI, the most notably is perhaps economics, where *homo economicus* is a term used to contrast the model of humans as rational self-interested actors with humans as they actually behave (*homo sapiens*). The rational agent model has been heavily criticized from both descriptive and normative angles.

Descriptively, many deviations from rational agency have been observed in humans, perhaps most famously by Daniel Kahneman and Amos Tversky [157]. Moreover, these deviations are not mere foibles peculiar to human beings. There are fundamental computational reasons why rational agency is impossible in our physical universe, and work on **bounded rationality** aims to account for these limitations [196].

Because the RL problem statement defines maximizing reward as its objective, RL also tacitly endorses rational agency as a normative standard. This corresponds to a consequentialist ethical stance, in contrast with alternatives such as deontology, virtue ethics, or commonsense morality, which may resemble each of these philosophical theories in various respects in different contexts [143].

Consequentialism often seems starkly at odds with commonsense morality, both in terms of the actions is allegedly recommends (e.g. killing an innocent patient so that their organs will save five others [265]) and the coldly calculating psychological

disposition it suggests, as typified by the rational – and, incidentally for our purposes, self-interested – "homo economicus". However, the thought experiments which philosophers have traditionally used to critique consequentialism may fail due to **moral cluelessness** [122], that is, our inability to predict the long-term consequences of different actions. Technically, *any* behavior can be viewed as maximizing *some* reward function; see the "simple existence proof" in Section 3.1 of Leike et al. [190]. Thus critiques of consequentialism in AI may need to develop more subtle arguments, e.g. about the 'naturalness' of expressing a given behavior as reward maximizing (or not).

Deontological constraints on an AI's behavior can be implemented in code via a conditional statement of the form "if the AI would perform a forbidden behavior, instead it performs [an allowed behavior]". However, imposing constraints during learning may be more challenging. Orseau and Armstrong [227] and Leike et al. [189] show that some RL algorithms may lead agents to seek to disable such constraints, when doing so increases their ability to collect reward.

Recent works also highlight the problem of **moral uncertainty** – that is, our inability to identify a single correct moral theory – arguing that it may need to be treated differently from other forms of uncertainty [206], suggesting possible implications for project of AI Alignment [78], and translating philosophical insights into RL algorithms [84]. Along similar lines, Eckersley [83] argues for viewing preferences as defining partial orderings, instead of the total orderings implicit in the use of reward functions. The argument rests on philosophical results about the impossibility of an ethical framework simultaneously assigning value to positive utility, negative utility and equality in accordance with widely held ethical intuitions; a classic utilitarian resolution to this dilemma is to reject that equality is intrinsically valuable, and directly compare negative and positive utility.

1.2 AI x-risk

1.2.1 Why would progress in artificial intelligence increase existential risk?

Recall that existential risk (x-risk) refers to the risk of humanity going extinct. Here we outline an argument that advanced AI systems themselves are likely to pose an existential

risk, based on three key concepts, namely 1) instrumental goals, 2) the difficulty of specification, and 3) safety-performance trade-offs. Note that a number of other arguments have been made for considering AI as an x-risk, or more generally, as contributing to x-risk; notable *recent* examples of writing making the case of AI x-risk include [45, 60, 222] I focus on this line of argument because I personally find it the most distinctive and compelling.

1.2.1.1 Instrumental Goals

Our description of reinforcement learning provides a good basis for understanding one of the most classic arguments for AI x-risk, called **instrumental goals** [22, 33]. Omohundro [225] introduces this concept (under the name "basic AI drives") in order to argue that programming an AI system to pursue a harmless "terminal" goal (e.g. winning a game of chess) will not necessarily lead to harmless behavior, and that instead, even such a system will seek to protect itself and acquire resources. These are examples of *instrumental* goals – self-preservation and resource acquisition seem broadly useful for achieving most ends; for instance, if the chess-playing AI were able to turn the entire solar system into a big computer that did nothing but imagine possible chess-moves, that would probably increase its chances of victory. But this same line or argument applies to any system that is trying to plan how to achieve some goal; the more resources it has to do the planning, the better.

In reinforcement learning, optimal behavior typically requires delaying gratification, that is sacrificing reward in the present in order to achieve more reward in the future. An example is the well-known **explore-exploit trade-off**: At any time, an agent can either choose the action it believes to be the best (exploit), or it can instead act to gather more information in order to make better decisions in the future (explore). Exploration in RL is an example of an instrumental goal, because knowing more is not intrinsically valued, it is *only* valued when it helps increase (expected) future reward.

All instrumental goals basically amount to the goal of having power, that is, the ability to control the physical world in order to direct events and thereby maximize the probability of achieving one's goals, or the extent to which they are achieved (e.g. the

discounted sum of future rewards. While an AI having power or influence over the future is not necessarily dangerous, its goals might have to be very similar to ours in order to prevent an existential catastrophe. For instance, an AI that did not care about maintaining humans' influence over the future would naturally come into conflict with humans, since we do care about our ability to control the future, and so are liable to disempower the AI, thwarting its ability to achieve its own goals.

Turner et al. [288] formalize power-seeking behavior in the context of MDPs, and show that optimal policies tend to seek power under fairly general conditions. As a caveat: whether it makes more sense to pursue instrumental goals or directly optimize the terminal goal depends on the extent to which an AI system delays gratification. A myopic AI system might be entirely free of instrumental goals.

1.2.1.2 Goodhart's Law and the Difficulty of Specification

Specification refers to the process by which we define and communicate our preferences to AI systems. The difficulty of translating our preferences directly into code should be immediately apparent. Rather, it seems necessary to employ learning in the specification process, just as learning seems necessary for many AI tasks. We can view Machine Learning approaches as implicitly specifying tasks such as "learn to classify images the same way a person would" or "maximize your score in this game". However, there is mounting evidence that specification is non-trivial, even for tasks like classification. Ilyas et al. [154] show that image classification is underspecified in practice – there are multiple ways of getting good (i.i.d.) test performance, some of which rely on very different features than humans do to make classification decisions. D'Amour et al. [66] similarly argue that underspecification is a common and critical problem.

Goodhart's Law is an adage, commonly stated as "When a measure becomes a target, it ceases to be a good measure" [308]. As discussed in Section 1.1.1, Machine Learning typically seeks to encourage desirable behavior in AI systems by optimizing some measure of performance (e.g. a loss function or a reward function). In practice, these measures are imperfect proxies. It might seem like good performance according to an imperfect proxy should simply be imperfect performance according to an ideal

performance metric, but this is often not the case; behavior optimized for a proxy can be very different – even catastrophically different – from intended behavior. We call such outcomes **perverse instantiations**.

Goodhart's Law has reared its head repeatedly in the development of AI. Here we review a few examples; Krakovna et al. [172] maintains a list of dozens.

1. **Boat Race:** In perhaps the most famous recent example, a boat racing video-game rewarded points for crossing checkpoints as a way to measure progress around the track. An RL agent playing the game found that it could cross the same checkpoint repeatedly (while repeatedly crashing), scoring many points without making any progress in the race [58].

2. **Evolved Creatures Falling:** An evolutionary algorithm searching for simulated creatures that could run fast was scored based on how far from the starting point the creature's body extended. One solution it found was a very tall creature that simply fell over [263].

3. **BLEU score:** Progress in translation and other NLP tasks has often used the BLEU score [231], which measures overlap in various n-grams between source and target sentences. Degenerate source sentences can get a high score [270], and a high quality paraphrases can get a low score [320]. Overall, scores are not highly predictive of human evaluations [252].

4. **i.i.d. accuracy:** In computer vision, super-human performance on ImageNet [174] has failed to translate into systems that work reliably on (even slightly) wider distributions of real-world images [241].

Why, besides examples such as these, should we expect Goodhart's Law to hold? Garrabrant [101] provides a taxonomy of four mechanisms for Goodhart's Law, i.e. reasons the relationship between the true objective U and a proxy V can fail To hold. These are depicted in Figure 1.1, which is taken directly from Garrabrant [101]. See also Manheim and Garrabrant [210] for similar content in the format of a technical report.

Quick Reference

- Regressional Goodhart - When selecting for a proxy measure, you select not only for the true goal, but also for the difference between the proxy and the goal.
  - Model: When U is equal to $V + X$, where X is some noise, a point with a large U value will likely have a large V value, but also a large X value. Thus, when U is large, you can expect V to be predictably smaller than U.
  - *Example: height is correlated with basketball ability, and does actually directly help, but the best player is only 6'3", and a random 7' person in their 20s would probably not be as good*
- Causal Goodhart - When there is a non-causal correlation between the proxy and the goal, intervening on the proxy may fail to intervene on the goal.
  - Model: If V causes U (or if V and U are both caused by some third thing), then a correlation between V and U may be observed. However, when you intervene to increase U through some mechanism that does not involve V, you will fail to also increase V.
  - *Example: someone who wishes to be taller might observe that height is correlated with basketball skill and decide to start practicing basketball.*
- Extremal Goodhart - Worlds in which the proxy takes an extreme value may be very different from the ordinary worlds in which the correlation between the proxy and the goal was observed.
  - Model: Patterns tend to break at simple joints. One simple subset of worlds is those worlds in which U is very large. Thus, a strong correlation between U and V observed for naturally occuring U values may not transfer to worlds in which U is very large. Further, since there may be relatively few naturally occuring worlds in which U is very large, extremely large U may coincide with small V values without breaking the statistical correlation.
  - *Example: the tallest person on record, Robert Wadlow, was 8'11" (2.72m). He grew to that height because of a pituitary disorder, he would have struggled to play basketball because he "required leg braces to walk and had little feeling in his legs and feet."*
- Adversarial Goodhart - When you optimize for a proxy, you provide an incentive for adversaries to correlate their goal with your proxy, thus destroying the correlation with your goal.
  - Model: Consider an agent A with some different goal W. Since they depend on common resources, W and V are naturally opposed. If you optimize U as a proxy for V, and A knows this, A is incentivized to make large U values coincide with large W values, thus stopping them from coinciding with large V values.
  - *Example: aspiring NBA players might just lie about their height.*

Figure 1.1: A list of mechanistic reasons why optimizing a proxy V can fail to optimize a proxy U, taken verbatim from Garrabrant [101].

The point of Goodhart's Law in the context of AI alignment is that we may need to specify the behaviour we desire from an AI system quite precisely. At the moment we do not seem to have reliable methods of doing so, as the examples above indicate.

The methods of specification common in AI also seem inadequate, conceptually. Reward functions are difficult to construct for simple robotics tasks [169]. Yet for more capable and powerful AI agents, reward functions might need to encode much more complex preferences and values, in order to avoid perverse instantiations. On the other hand, (un)supervised learning primarily optimises for performance on the training set, with only very weak selection for (aligned) generalisation, typically via early stopping. In either case, the amount of information about our preferences that is transmitted to the AI system seems woefully inadequate.

The underlying issue seems to be our inability to formalise or otherwise convey complex human concepts. Specification could be much easier if we shared a language with AI systems. Having a shared language really amounts to 2 things: (1) having a shared vocabulary of concepts, and (2) having a mutually understood way of referring to different concepts. This is why getting AI systems to understand human concepts seems critical for specification; it addresses (1).

Note, however, that it does not necessarily address (2): an AI system could understand human concepts, but reference them in different ways. For instance, it might understand what cats are, but it might nonetheless behave as if the word or symbol for "cat" means "cat that is not in an unusual pose, location, or situation; or alternatively, any of the various objects or trappings most commonly associated with cats". Mordvintsev et al. [217] and Beery et al. [16] present examples of machine learning methods suffering from these kinds of confusions: dumbbells seem intrinsically coupled to muscular forearms in Mordvintsev et al. [217], and cows on a beach are not recognized as cows in Beery et al. [16]. It is unclear if the source of these particular problems is (2) or (1), however.

So how can we get AI systems to understand human concepts? And can we train AI systems to understand natural language? Currently the most promising approach to teaching AI systems human concepts and human language seems to be unsupervised learning. In unsupervised learning, we do not provide *any* explicit information about our

preferences. The objectives used in training these models are clearly misaligned with most of the ways in which we'd like to leverage their capabilities.

Unsupervised learning is typically complemented with post-hoc efforts to align the model, e.g. via further training known as **fine-tuning**, to direct it to solve the tasks we are actually interested in. This paradigm has been quite successful, but if we view specification as a matter of communicating sufficiently specific information about our preferences to AI systems, this success seems somewhat tenuous and suspect. The amount of explicit information about human preferences provided is typically orders of magnitude less than what is used for unsupervised training, and does not seem to be sufficient to specify our preferences with enough precision, unless the AI already has a very strong prior over our possible preferences. Moreover, available compute and unlabeled data grow much faster over time than data about human preferences does.

Still, it does seem plausible that this approach – unsupervised learning + post-hoc alignment – *could* work, at least in some circumstances. For instance, consider training a model using next-step prediction on human-generated text containing examples the kinds of natural language-based tasks that we might want the model to perform. Then this model might already have a good understanding of what the tasks are that we might want it to perform, and only need a small amount of information to determine which task is currently desired. In this example, the training data being generated by human behavior seems important; it is not clear if such reasoning would generalize to a system trained on different types of data, e.g. from sensors in the physical world.

A more fundamental and philosophical difficulty for specification is that the idea of specifying and optimizing some performance measure is itself a leaky abstraction. One way in which this abstraction can break down is **wireheading**; an example of wireheading would be an AI systems directly manipulating the memory register where the measure of performance is stored. If it is stored in multiple locations, it's unclear which one the AI system would or 'should' end up caring about – this is the break-down of the abstraction. A similar example that doesn't require manipulating computer memory would be to manipulate sensors, for instance pointing a camera at a video stream recording a highly rewarding event. While it might be easy enough for a reasonably intelligent AI system to

realise that wireheading is not the behaviour the user intends, this does not mean that it will be incentivised not to wirehead. In this sense, wireheading is like an extreme version of Goodhart's law, where *any* specification actually boils down to 'hijack the memory where your performance metric is stored', for an AI system that is able to realize that goal.

We might hope to address this issue by using human feedback to punish wireheading. However, this creates an incentive for the AI system to hijack the human feedback signal, e.g. by manipulating again its hardware, or also/instead by manipulating the human. Manipulating the human could look like coercion, direct physical intervention to control the human's body, or more subtle forms of influence. Ultimately, without a clear notion of what constitutes manipulation, or some way of removing the incentive for manipulation (see Section 1.3.6.3), it seems we are left back in the position of being unable to prevent wireheading.

Wireheading might seem to be harmless, conjuring the image of the AI as an addict content to ignore the external state of the world and "get high" on its reward signal. But this sort of addictive behaviour in humans can be seen as resulting from a failure to delay gratification. An AI system with the goal of maximising its long-term reward would have a powerful incentive to protect its ability to wirehead, leading it to pursue the same instrumental goals as a system whose terminal goal is to influence the world in a more impactful way.

These arguments about the difficulty of specification are not meant to be conclusive. But they do provide a basis for concern. Misspecification and underspecification may not lead to perverse instantiation in practice, they may simply lead to poor performance. However, for AI agents with long-term goals, it does seem necessary to ensure that a specification precludes dangerous behaviours resulting from the pursuit of instrumental goals. An AI system that attempts to remain under human control is sometimes called **corrigible** [57, 267]. Corrigibility seems like a plausible minimal requirement for x-safety for such systems. Corrigibility may require somewhat sophisticated understanding of human psychology in order to avoid human manipulation, and misspecification could lead to failures of corrigibility.

Overall, I find the arguments for the difficulty of specification and the dangerousness of instrumental goals quite compelling, *for AI agents with long-term goals*. But can we not simply avoid building such AI systems, if they prove too dangerous? I will now turn to addressing this objection.

1.2.1.3 Social Factors and Safety-Performance Trade-offs

The arguments of the previous two sections establish reasons that AI agents with long-term goals might pose an existential risk to humanity. In this section I argue that humans are likely to deploy such AI systems despite this risk.

The basic argument is that such systems will offer significant performance benefits and that competition between different humans and human organisations will drive a race-to-the-bottom dynamic, wherein increasing levels of x-risk are tolerated by individual actors. The level of x-risk that an individual actor will tolerate will also be higher than the socially optimal level of x-risk, because x-safety is a common good.

This argument relies on the premise that there is a trade-off between safety and performance of AI systems. One way to trade-off between safety and performance, given the arguments from the previous sections, is to modulate how far in advance an AI system plans. Planning further into the future makes an AI system less safe because it is more likely to pursue dangerous instrumental goals, as argued in Section 1.2.1.1. Even if its terminal goals are reasonably well aligned with ours, we can still expect a significant risk of perverse instantiation based on the arguments of Section 1.2.1.2.

1. **Adding constraints** would limit the ability of an AI system to pursue its goals effectively.

2. **Doing more safety testing** before deploying a system would reduce ones access to cutting edge capabilities.

3. **Increasing interpretability** seems likely to trade-off against its performance; one reason for this is that intelligent behavior is inherently quite complex and difficult to summarize, and so places greater demands on interpretability. For examples, decisions made by Deep Learning systems are currently not well understood.

4. **Keeping a human-in-the-loop** would limit the reaction time of an AI system and might make it less competitive in dogfights, for instance [67].

5. **Limiting the sensors and actuators** an AI has access to would give it less situational awareness and less ability to directly enact its plans.

6. **Training offline** prevents an AI system from directly enacting plans, allowing human overseers to run safety tests before the system is deployed. But offline training also limits an AIs ability to efficiently run 'experiments', and collect the most valuable data, and build a causally accurate model of the world.

This list is not exhaustive. Given the many axes on which actors can trade-off safety and performance, it seems likely that safety-performance trade-offs will be extreme. An AI developer that puts a premium on safety will struggle to compete with developers that take more risks.

While it might seem far-fetched that such a race-to-the-bottom would actually occur, the stakes of winning an AI-driven competition will increase as AI systems become more powerful. Ultimately, with the development of AGI and more advanced AI systems, the entire geopolitical fate of different countries might hang in the balance of AI-driven economic, military, and/or intelligence competitions (e.g.). Controlling such unbridled international competition seems incredibly difficult, given the current geopolitical world-order.

1.2.2 On the desirability of existential safety

Tegmark [280] reports a conversation between Google founder Larry Page and OpenAI founder Elon Musk, in which Page argues that reducing AI x-risk might be "speciesist", since digital life is "the natural and desirable next step in the cosmic evolution". Here, I will mount a brief defense of existential safety as desirable, even if one shares the transhumanist ethical inclinations of Page. This defense is comprised of four arguments:

1. Even if we think AI/AGI should be accorded ethical status similar to or greater than humans (either as moral agents and/or moral patients), this does not absolve us from considering the ethical implications of our design choices (by analogy with humans, e.g. genetically modifying people to be evil and/or to increase their suffering seems bad).

2. We should seek to solve the Pretty-Hard Problem of Consciousness [1] before accepting an outcome where AI systems lead to human extinction.

3. It is probably worth expending significant effort now to try to decrease our chances of making mistakes while creating AI systems capable of overpowering and replacing humans.

4. Pursuing a transhumanist future in the knowledge that it will lead to human extinction seems anti-social.

In total, the line of argument is: Even if biological humans ought to be replaced by some form of digital life, not all futures populated by digital life are equally good, and thus we should try to steer the future towards those that are more desirable (1), and try to exercise control over the future as long as we believe we have the agency and clarity to steer towards better futures (3). One specific way a future populated by digital life could go wrong is if those life-forms are conscious and have overwhelmingly negative subjective experiences; their having no subjective experience at all would also be catastrophic under many world-views (2). Finally, many people would view the extinction of biological humans as a catastrophe. Even if one believes this view to be in error, failing to account for others' preferences would seem to be an act of epistemological arrogance, and precipitating human extinction solely on the basis of one's own personal views leaves one vulnerable to the Unilateralist's curse [35] (4).

1.2.3 A Brief and Incomplete History of AI x-risk

Terms in recent use for describing concerns related to AI x-risk include:

- friendly AI

- beneficial AI
- the AI control problem
- AI safety
- AI alignment
- AI x-risk
- Human-Compatible AI

Until recently, "AI safety" was the most commonly used term for technical work on reducing AI x-risk. However, "AI Alignment" is now more popular, and is more specific to x-safety (as opposed to safety more broadly).

As mentioned earlier, pioneers in AI were concerned that it might lead humans to lose control over the future. However, Nick Bostrom seems to have been the first academic to do serious work addressing this concern, beginning with his first publication "How Long Until Superintelligence" in 1998 [31].

Bostrom's Future of Humanity Institute (FHI) was founded in 2005, the same year in which the Machine Intelligence Research Institute (MIRI) pivoted from trying to accelerate the development of AI to focusing on AI x-risk. MIRI was founded by Eliezer Yudkowsky – who somewhat notoriously lacks academic credentials – in 2000.

A pivotal moment arrived with the publication of Bostrom's *Superintelligence: Paths, Dangers, Strategies* [34] in 2014 which led to broader public awareness of the types of concerns mentioned in Section 1.2. Up to this point, the resurgence of interest in AI x-risk seems to have developed primarily outside the mainstream AI/ML research communities, with the exception of some interaction with the community of researchers explicitly working on AGI, e.g. participants at the *Conference on Artificial General Intelligence*, held annually since 2008.

Superintelligence arrived in a media climate already full of AI hype because of the recent successes of Deep Learning, most notably AlexNet [174], but also Deep Q-Learning [214]. Popular articles about AI x-risk featured photos from the Terminator franchise, and prompted a significant backlash among ML researchers.

Prior to the advent of Deep Learning, researchers reportedly had great difficulty publishing work on neural networks, and discussion of AGI was outside the intellectual Overton window of the ML community. The *Conference on Artificial General Intelligence*, which was founded in 2008, appears to have been something of a reaction to the broader field's lack of interest in AGI. Its inaugural proceedings included the seminal work of Omohundro [225] on the instrumental convergence thesis (see Section 1.2.1.1).

Already in 2009, Russell and Norvig [249] included discussion of AI x-risk, and Omohundro [225] in particular, in their preeminant AI textbook. Since then, Stuart Russell has become the primary academic champion for addressing AI x-risk, leading the creation of the Center for Human-Compatible AI (CHAI) at the University of Berkeley which focuses on relevant research. Russel and collaborators also developed the framework of **assistance games** [129, 258], in which an AI system aims to optimize a human overseer's reward function, which is treated as a latent variable. This incentivizes the AI system to manage uncertainty about human objectives appropriately, although assistant games still require specifying a prior over human reward functions, and the conditions under which the prior leads to aligned behavior in practice are not known. Most recently, Stuart Russell published a popular science book on AI x-risk, alignment, and assistance games [248].

Another important individual to this history is Paul Christiano. While Bostrom and Yudkowsky often focus on reasons for pessimism about ideas for aligning AI, Christiano has provided the community with more reasons for optimism. This includes optimism in approaches rooted in Machine Learning, whereas Yudkowsky has emphasized the need for further work on **agent foundations**, in order to understand principles which might govern super-intelligent reasoning. For instance, Christiano has argued that solving the full value-loading problem (i.e. getting an AI system to understand all of human values) is likely unnecessary, and intent alignment (see Section 1.3.2) is likely sufficient and achievable. The optimism/pessimism epitomized by Christiano/Yudkowsky have been a recurring thread in informal discussions in the alignment research community; such disagreements seem to mostly hinge on intuitions that are difficult to precisely and exhaustively articulate.

The publication of "Concrete Problems in AI Safety" [5] was another landmark. This work helped popularize concerns about AI safety within the Machine Learning community, and helped refute the perception that work on AI x-risk was inherently philosophical in nature.

Other significant developments were the establishment of DeepMind (in 2010) and OpenAI (in 2015). These world-leading AI research organizations both made AI safety a core part of their stated mission. A number of other organizations played a significant role in building the field of AI safety/alignment. The Future of Life Institute (FLI) is noteworthy for their efforts to fund AI safety research (with the financial support of Elon Musk, and more recently, Vitalik Buterin), as well as their community building efforts, which brought together leaders of the field, donors, and thought leaders for several conferences on benefitial AGI, and produced a letter signed by many AI luminaries about risks of AI, including the risk of loss of human control [95].

Also relevant is the development of concerns about social impacts of AI outside of the x-safety community. These include concerns such as:

- Fairness, e.g. racism, sexism, ableism, ageism, and other forms of AI-enabled algorithmic discrimination.
- Economic impacts of AI, e.g. automation leading to technological unemployment and extreme concentrations of wealth, the monopoly power of big tech/AI companies, the importance of data in AI and the failure of existing social/economic models to compensate data providers.
- Privacy, e.g. the propensity of AI systems to leak private information from their training data.
- Near-term safety, e.g. the safety of self-driving cars, or AI-powered infrastructure.
- Military use of AI, e.g. lethal autonomous weapons.
- Environmental impacts of AI systems, e.g. the rapidly increasing energy-use of AI systems.

- Other ethical concerns about AI, e.g. where to place responsibility for AI misbehavior, and how AI systems should be programmed to address ethical dilemmas.
- Other social impacts of AI, e.g. AI-powered feedback loops in school rankings, predictive policing algorithms, or content recommendation engines.

These concerns have some-times been referred to as "near-term" in contrast with the "long-term" concern of AI x-risk. While this distinction can be useful, as these concerns are more universally acknowledged than AI x-risk, I consider this language misleading and divisive. A common objection to concerns about AI x-risk is that it is a distraction from (e.g. some of) these concerns, which are more important, pressing, and/or more realistic [61]. I believe that historically exactly the opposite is true, and that attention for any concern around the social impacts of AI has tended to lead to more attention to all of the others as well. Furthermore, I believe there is also considerable overlap in terms of the steps needed to address these various concerns. There is a need for better understanding of how to responsibly develop and deploy AI systems, and for widely adopted norms, standards, policies, regulations, and/or laws, etc. to ensure responsible AI development and deployment. There are fundamental technical and governance challenges that these needs raise, and addressing them should be a common objective for communities working on x-safety and other social impacts of AI. *The Alignment Problem* [54] connects existing social impacts of AI and AI x-risk under the unifying theme of AI Alignment.

1.2.4 How can we increase our existential safety?

Suppose one accepts that AI poses an existential risk, and feels (e.g. morally) compelled to do something. What can be done? The central argument for AI x-risk presented in Section 1.2 smacks of inevitability. Competitive social pressures will drive us to develop increasingly advanced AI, even if they increase x-risk. And more advanced AI systems seem destined to be goal-directed, with attendant instrumental goals. Perhaps we will be able to program AI systems to pursue our goals, but existing approaches to specification don't seem sufficient.

The "global coordination" solution. Perhaps the most obvious solution would be for all of humanity to coordinate to develop AI responsibly, and in particular, to avoid building AI systems that pose too much x-risk. We could instead make decisions about how to develop AI on the basis of the common good, preventing individual risk-taking from raising (estimated) AI x-risk beyond some agreed on level, e.g. 1% per century. It is hard to imagine what such a solution would look like in practice, but it would likely accompany a much greater ability to address other global coordination problems (such as climate change).

The "technical research" solution. On the other hand, we might imagine a primarily technical solution. If our primary concern is AI systems that do not behave as intended, then we can imagine finding ways to ensure AI systems act in accordance with our intentions, this is sometimes called **AI Alignment**. We discuss AI Alignment and relevant technical areas of AI research in Section 1.3.

The above approaches are complementary. Effective global coordination would allow us to enforce a set of safety standards prohibiting the development of certain AI systems deemed to pose too much x-risk. Meanwhile, technical research can expand the kinds of systems which can be safely developed. Advances in coordination or technical work can substitute for each other to some extent. If technical advances make fewer AI systems risky to develop, then it should be easier to enforce standards. On the other hand, even if no technical progress was made, we might be able to coordinate to prevent developing AI systems more dangerous than those currently deployed, e.g. by prohibiting AI research and development entirely until suitable safety procedures could be developed. In the most extreme case, this could mean forgoing or delaying many applications of AI that could have huge positive impacts – for example, curing many existing diseases, or reducing x-risk from other sources.

Technical work and coordination cannot always be neatly distinguished, and it may be necessary to consider them as more of a continuum. Global coordination can itself be approached via technical work; for instance, the field of **mechanism design** (sometimes called "reverse game theory") studies how to create games where players' incentives align with some notion of collective good. At the same time, technical work that does

not focus on coordination problems might still need to consider how multiple AI systems will interact with each other, humans, and human institutions.

At a more fundamental level, thoroughly addressing AI x-safety would likely involve a broad multi-disciplinary research effort and substantial social change. Historically, AI has primarily been developed in the model of a single rational agent. But human intelligence and values both appear to have a large social component. Technical work on reducing AI x-risk, and in the field of AI as a whole, often assumes away the additional complexities this raises. Several salient questions in my mind are:

1. **How should we aggregate decisions from group members?** What are the practical implications of impossibility theorems, such as Gibbard's theorem [106]? Which approaches to group decision-making tend to lead to results group members are satisfied with?

2. **Does it make sense to think of individuals as the nexus of decision-making?** Can we move beyond individualistic conceptions of self, identity, and agency? What would that look like?

3. **Does it make sense to talk about individuals having preferences?** If so, what mathematical form do they take (e.g. partial or total ordering, cardinal or ordinal utility function)?

4. **How can we elicit preferences effectively?** When can stated or revealed preferences be relied on? What sort of human-modelling is required in order to understand how our behavior reveals our preferences, values, subjective experience, etc.?

The evident limitations of viewing humans as rational agents suggests that AGI systems might not be rational agents either. One could instead take a view of the world more in-line with Foucault's views on power, which he often seems grant a causative force akin to agency. Summarizing his views is beyond our scope, but this quote provides a characterization that is good enough for our purposes:

> His work marks a radical departure from previous modes of conceiving power and cannot be easily integrated with previous ideas, as power is diffuse rather

> than concentrated, embodied and enacted rather than possessed, discursive rather than purely coercive, and constitutes agents rather than being deployed by them [238].

Alexander [3] provides a similar perspective on the personification of Moloch in Ginsberg's poem "Howl", describing a "conception of civilization as an individual entity" who is attributed agency ("*Moloch does it*"), and is sometimes erroneously conflated with capitalism.

To the extent that such views – which do not center rational agents as the nexus of agency – provide a better description of reality, they may seem to weaken the case for AI as an x-risk, especially the arguments of Section 1.2. However, there are a number of reasons to take such arguments seriously regardless. First, rational agency is not an absolute, and in particular human behavior arguably still approximates rational agency in many ways in many circumstances. Second, a number of limitations on human rationality can be explained as resulting from algorithmic or computational limitations, some of which advanced AI systems might surpass. Furthermore, current AI systems are overwhelmingly developed within a framework of rational agency. Finally, it is worth noting that even if individuals are not well-modeled as rational agents, there may still be an appropriate analysis, e.g. at the level of groups, genes, memes, etc., that does identify agent-like forces at play. For instance, an AI-driven economy that effectively ends up optimizing for growth might pose as much x-risk as a single superintelligent AI agent. Critch [62] introduces the related concept of a "Robust Agent-Agnostic Process (RAAP)", which may or may not be agent-like, but importantly, is robust to attempts by individual agents to disrupt it. He also explains how RAAPs could induce AI x-risk, and identifies existing thought on AI x-risk (e.g. AI "arms-races") that can be viewed through this lens. We discussed the related issue of "multi-multi delegation" in Critch and Krueger [63], and argued that the emergent dynamics of multiple humans trying to delegate work to multiple AI systems pose additional challenges. Overall, I find that arguments for AI x-risk arising from AI agents can typically be adapted or elaborated to show how similar – and sometimes more severe – concerns arise in contexts where threats are not concentrated in a single, identifiable AI agent. Nonetheless, our approach to AI x-safety

should be robust across these scenarios.

In summary, increasing AI x-safety can be crudely decomposed into technical work aimed at increasing our ability to control and direct (collections of) AI systems towards desirable ends, and governance work on improving our ability to coordinate to ensure that AI is developed responsibly. However, effective work might sometimes be difficult to categorize, and this distinction might sometimes be conceptually limiting and even impede progress. Having acknowledged these caveats, in Section 1.3 I will argue that narrowly scoped technical work on the AI Alignment problem may have the potential to significantly reduce AI x-risk, or at least provide insights that help motivate, mobilize, and clarify directions for further work on AI x-safety.

1.3 AI Alignment

A number of different terms have been used to describe technical work aimed at increasing AI x-safety, and the community engaged in such work. There is not a mature sub-field of AI research with this focus, and many existing areas are relevant. But recently, researchers have *mostly* settled on the term "AI Alignment" to describe this project. AI Alignment can be thought of as encompassing all such work, or as tackling the more narrow topic of 'how to get a single AI system to do what a single user wants it to do'. I will use it in this second sense, since I believe this provides a useful scope, despite the limitations mentioned in Section 1.2.4. In particular, it is largely agnostic towards the "preference payload" [190], that is the question of *whose* intentions the AI system should adhere to. This is obviously a critically important question, but not a technical one.

1.3.1 Alignment Targets.

But note that (what I call) AI Alignment *is* always considered relative to some "user", which might be a single human, but could also be any number of other things. Potential users, or types of users, are sometimes called **alignment targets**. Proposing an alignment target amounts to explaining or outlining a process for generating a (hopefully coherent) set of preferences. Examples of alignment targets include:

1. **Revealed preferences of a single human**. We might simply assume that a person's choices are an accurate reflection of their "true" preferences. This seems problematic, however, because of cognitive biases and bounded rationality.

2. **A single human who thinks for infinitely/arbitrarily long**. We might imagine that we could make better decisions, which more accurately reflect 'what we really want', if we had more time to deliberate.

3. **Human Consulting HCH (HCH)**[3] [55, 56]. Like the previous example, HCH tries to capture "a human's enlightened judgement" [55]. It can be pictured as an infinitely/arbitrarily deep tree of copies of that human, able to delegate aspects of decision-making to the sub-trees below them.

4. **Coherent Extrapolated Volition (CEV)** [315] is an outline for how we might think about something like *humanity*'s enlightened judgement. Quoting Yudkowsky [315], "In poetic terms, our *coherent extrapolated volition* is our wish if we knew more, thought faster, were more the people we wished we were, had grown up farther together; where the extrapolation converges rather than diverges, where our wishes cohere rather than interfere; extrapolated as we wish that extrapolated, interpreted as we wish that interpreted."

There are clearly philosophical aspects to selecting an alignment target. There are also more practical questions about how to implement different targets, and how to increase our confidence in our implementations. For instance, we might try to approximate (2) by training an AI system to predict $H(Q,T)$, that is, the response of a human, H, to a query, Q, given T seconds to think about it, and then hoping it can extrapolate to much larger T. To believe that such a scheme has any chance of working we would need to trust our model to extrapolate well. And we would not be able to confirm that our trust was well founded, since we would not be able to realize the alignment target, only schemes to approximate it. It seems difficult to achieve a high level of justified confidence in such schemes, but doing do could be crucial for AI x-safety.

[3] This is a recursive acronym.

1.3.2 Alignment vs. Capabilities; Intent Alignment

A popular, informal, distinction is between the alignment and capabilities of an AI system. 'Capabilities' refers to the general level of competency and/or intelligence of an AI system, e.g. How well does it understand the world? How effectively can it plan to achieve various goals? Which types of interesting behaviors could it perform, if properly motivated? Here, 'Alignment' refers to how well the AI's motivations match the overseer's intentions or preferences. This notion of alignment, sometimes called "intent alignment" differs from how I (mostly) use the term in this thesis. Whereas Alignment is about 'doing what a user wants' Intent Alignment is about '*trying* to do what a user wants'.

In this framing, for an AI system to do what we want it has to both understand what we want and how to achieve it (capabilities), and to *want* to do what we want (alignment). This type of language may make more or less sense in different contexts or when speaking about different AI systems. For instance, consider an AI system which is a mapping from reward functions to policies. How well these policies optimize the corresponding reward functions constitutes this system's capabilities. For a given reward function, the alignment of the system is how well that reward function matches the user's true preferences. A similar example can be constructed using models whose behavior is conditioned on natural language inputs, such as GPT-3 [39]. To my knowledge, a more rigorous or formal distinction between alignment and capabilities has so far remained elusive.

1.3.3 Decomposing AI Alignment; relevant research areas in machine learning

There have been a number of attempts to decompose the Alignment problem or enumerate useful directions for research. I won't provide an overview of these works here; readers can refer to Critch and Krueger [63] for a review of several other research agendas, and Ortega et al. [228] for more resources. Instead, I will summarize and comment on the decomposition I've found most useful: Specification, Robustness, and Assurance [228]. And I will provide a (non-exhaustive) list of areas of AI research that could be useful for AI Alignment.

Specification refers to defining what we want an AI system to do and communicating that information effectively to the AI system. The difficulty of specifying what we want in terms an AI can understand is one of the classic causes for concern about AI x-risk (see Section 1.2.1.2). **Robustness** is about making a system less vulnerable to unexpected challenges, such as encountering new situations. AI systems are notoriously brittle, e.g. manufacturing robots are still mostly confined to carefully controlled production lines. **Assurance** is about how we can increase our justified confidence that running an AI system will not lead to alignment failures. Even if an AI system behaves exactly as we intend (i.e. perfect specification) in any circumstance (i.e. perfect robustness), we would still like to *know* that this is the case (i.e. perfect assurance).

Examples of machine learning research areas helpful for **specification** include:

- **Methods of learning reward functions** such as reward modelling, Inverse Reinforcement Learning, assistance games, and active reinforcement learning are straightforwardly relevant to specification; a reward function is a natural way to encode a specification.

- **Learning generic human preferences** such as widely held human values, or expectations humans might place on AI systems, could make specifying specific tasks much easier by removing the need to explicitly specify all of the things that a human assigned that task would typically take for granted (e.g. don't kill anyone, don't break anything, ask for clarification if needed, avoid irreversible side effects).

- **Constraints** and methods for handling them could be useful for encode generic/background human preferences, and for specifying unacceptable behaviors.

- **Incentive Management Techniques** can be used to specify which forms of deliberate influence are deemed acceptable.

- **Scaling human feedback** could be important in order to increase the amount of information we can provide about our preferences. Semantic loss functions, (inter)active learning schemes, and augmented human judgments are example approaches.

- **Prompt engineering** and other methods of aligning large pre-trained language models fall into the specification category.
- **Multi-objective optimization** allow to postpone decisions about how to trade-off competing values until available options are determined.

Examples of machine learning research areas helpful for **robustness** include:

- **adversarial robustness** can help improve worst-case robustness
- **non-adversarial robustness** is more targeted at average-case robustness
- **domain generalization and domain adaptation** methods aim to improve models' robustness in entirely novel settings
- **Out-of-Distribution detection** can help AI systems to fail gracefully, given a safe backup plans.

Examples of machine learning research areas helpful for **assurance** include:

- **theory of learning, generalization, etc. (including empirical work)** can help designers understand general principles of AI systems, helping them predict their behavior.
- **interpretability** can help designers understand specific AI systems and predict or understand their behavior.
- **agent foundations** is a specific area of theory aimed at understanding fundamental aspects of rational agency, including the limits of existing conceptions of rational agency.
- **formal verification** methods can provide provable guarantees of well-defined specifications.
- **safe exploration** methods could help ensure that AI systems can be trained without failing catastrophically.

- **offline training** can prevent catastrophic failures during training, by training the agent in a safe environment (e.g. in simulation, a controlled lab setting, or entirely based on pre-collected data).
- **analyzing incentives** can help determine whether particular forms of failures, e.g. pursuing dangerous instrumental goals, are likely to occur in a given training/deployment scenario.
- **testing, e.g. unit tests** can be used to collect empirical evidence about how agents will behave in a variety of scenarios they might encounter.

1.3.4 The relationship between AI Alignment and AI x-risk

Alignment failures are not the only source of AI x-risk. As mentioned in Section 1.2.1.3, x-safety is a common good, and so actors are likely to prefer an AI system that carries socially unacceptable levels of x-risk, if it offers some benefit over systems carrying less x-risk.

Nonetheless, Alignment failures still seem like an important source of AI x-risk. When considering possible human extinction scenarios, Alignment failures and coordination failures often *both* play important roles, and are hard to disentangle. For instance, actors might underestimate the x-risk posed by a given system because they fail to notice a subtle alignment failure. Such failures seem more likely to be noticed in a less competitive environment where there is less cost to performing further safety checks. Better techniques for aligning AI systems, and a better understanding of possible failure modes seem valuable in such scenarios, since they could reduce the probability of such oversights.

Furthermore, aligned AI systems can help address global coordination problems, such as x-risk, for instance by helping propose and implement governance mechanisms. On the other hand, alignment techniques might also expand the range of tasks which AI systems can perform competently – without ensuring that they perform them safely. Misalignment seems likely to be a critical obstacle in getting AI systems to perform tasks

that are difficult to specify. If our ability to align AI systems increases smoothly, then the AI systems we build will pass through 3 stages of alignment:

1. **Insufficient Alignment:** AI does not understand the task well enough to appear useful.

2. **Apparent Alignment:** AI can perform the task well enough to appear useful, but is not actually useful (all things considered), because it suffers from critical undetected failure modes. These may be undetected because they are rare, long-term, subtle, or obscure.

3. **Sufficient Alignment:** AI can perform the task well enough to be genuinely useful; the benefits from using the AI to perform this task outweigh the costs of potential failure modes.

The ability to reliably distinguish between apparent and sufficient alignment would help prevent risks stemming from premature deployment of apparently-but-insufficiently aligned AI systems, thus making progress in alignment closer to an unmitigated good.

Whether a given level of *intent* alignment is sufficient might depend on the capabilities of the AI system in question. Consider the analogy with an anti-social human held in check (i.e. aligned) only by the fear of social sanction. If they were much more intelligent, they might be able to find ways of achieving their anti-social ends that avoid detection, and thus avoid social sanction. Thus they would be aligned at their current capability level, but not at a higher capability level.

1.3.5 Research Prioritization Through the Lens of AI x-safety

From the point of view of increasing x-safety, there are several reasons one might choose to do technical research:

1. To make progress on techniques that seem useful or necessary for increasing x-safety, such as AI Alignment.

2. To increase understanding of AI systems in a way that informs strategies for increasing AI x-safety, such as technical research priorities. This might include things like understanding the limitations of Deep Learning, or forecasting developments in AI.

3. To inform actors, e.g. researchers or policy-makers, and/or influence them towards taking effective action to reduce AI x-risk. This could include work that demonstrates novel or under-appreciated failure modes.

4. To build career capital and achieve a position for more effective advocacy or field-building. This could include work that demonstrates technical prowess, insight, engagement with social impacts of AI, or other individual virtues.

One method for deciding which particular research directions to pursue is the **importance, tractability, and neglectedness (ITN) framework** of Effective Altruism [85]. The goal of considering these three factors is roughly to maximize counter-factual impact by finding research that is actually likely to help reduce AI x-risk (important, tractable), and would not have been done otherwise (neglected). Neglectedness may be especially important in ML, as it is common for multiple researchers to be working on the same topic, "racing" each other to be the first to publish and reap the attendant recognition – and citations. This can be a useful strategy for building career capital (4), but seems harder to justify in terms of direct impact (1).

As reasoned in Section 1.2.1.3, AI x-safety may require that AI Alignment techniques stay "ahead of" AI capabilities, since increasing capabilities can put more demands on alignment. Towards this end, Bostrom [32] proposes **Differential Technological Development (DTD)**, i.e. aiming to push forward the development of technologies that will increase x-safety faster than those that increase x-risk. It can be difficult, however, to estimate the overall impact of a given project or area on Alignment vs. capabilities, respectively. And as argued in Section 1.2.4 Alignment itself might not reduce AI x-risk without the concurrent development of methods of global governance and assurance for AI systems. Despite such uncertainties, DTD seems like a useful lens through which to view these decisions.

A special, and perhaps non-obvious, case of DTD is attempting to hasten and/or steer the development of a research sub-field that would likely have developed soon anyways. For instance, Ian Goodfellow and Dan Hendrycks played a large role in "seeding" the sub-fields of adversarial and non-adversarial robustness (respectively) in deep learning. While this kind of contributions might seem to have limited counter-factual impact, as reasoned in the paragraph above, advancing the timeline on a research topic by a few months or years might still represent significant progress (1), especially given the pace of research. Moreover, while a given sub-field might be destined to emerge, *how* exactly that field operates, e.g. which benchmarks and methodologies are commonly used, or which problems are considered the most central, can determine how useful – or detrimental – progress in that field is for x-safety. For instance, it seems useful to focus attention on those Alignment techniques that are more likely to scale to AGI and beyond. As a specific example, Goodfellow [109] (a presentation) argues that the research community has been too focused on the formulation of adversarial robustness introduced in Goodfellow et al. [115], depriving adjacent concerns of the attention they deserve. Yet as of Dec 31, 2021, Goodfellow [110], the research agenda associated with Goodfellow [109], has only been cited 12 times; despite Ian Goodfellow's leading role in seeding adversarial robustness research, it seems his recent attempt to steer researchers towards the type of work he considers more relevant in hindsight has born little fruit. Brown et al. [38], Gilmer et al. [107] make similar calls to rethink and broaden the scope of adversarial robustness research.

1.3.6 A brief overview of my work on AI x-safety not included in this thesis

Here I'll describe 3 papers I contributed to that were directly motivated by AI x-safety.

1.3.6.1 ARCHES

In Critch and Krueger [63], we focus on the problem of AI x-risk, as opposed to the more narrow problem of AI Alignment. This work includes a large research agenda, but I will focus on summarizing several of the novel perspectives we advanced:

- **Explicit focus on AI x-safety:** Previous works have focused on safety, alignment, or beneficience. AI x-safety is a more specific goal than safety or alignment, and is more value-neutral than beneficience.
- **Prepotence instead of superintelligence:** We note that the key attributes of AI systems that induce x-risk are that they engender an unstoppable transformative impact that is beyond the impact of humanity in scale. While superintelligent AIs seem likely to be prepotent, they need not be. Likewise, AI that enables or powers unstoppable self-replicating "grey goo" would be prepotent, but need not be superintelligent.
- **Multi/Multi delegation instead of Single/single alignment:** Most technical work on AI x-risk has focused on aligning a single AI system with a single user. In contrast, we emphasize scenarios in which multiple AI systems are deployed by multiple actors, and the distinctive risks and research directions this perspective entails. Since results such as the impossibility theorems of social choice theory suggest that there is no 'correct' way to determine group preferences from individual preferences, the term "alignment" seems like a poor fit for this discussion, and we use "delegation" instead.

One contribution which I'm proud of is a thought experiment suggesting the inadequacy of single/single AI Alignment for reducing AI x-risk. I would summarize this as: 'Imagine that tomorrow everyone wakes up 1,000,000 time smarter' – this is meant as an analogy for (the sudden development of) a perfect solution to (single/single) AI Alignment. It is hard to predict how such a hypothetical would play out. But it seems highly plausible that existing social institutions that make destructive competitions less likely would break down, as more intelligent actors would rapidly find ways of subverting them. This might precipitate a successful power-grab by ambitious anti-social actors, or a descent into anarchy and unrestrained competition between actors.

1.3.6.2 Reward Modelling Agenda

Leike et al. [190] present an agenda that advocates for reward modelling – that is,

learning reward functions from human feedback – as an approach to solving specification problems. We discuss challenges for reward modelling and approaches to addressing them. We also propose "recursive reward modelling", an approach to specification which uses trained reward models as aids for human judgment, and which resembles HCH (Section 1.3.1). One notable result is a simple existence proof demonstrating that any policy can be uniquely specified by the appropriate choice of reward function.

A contribution which I'm proud of is our emphasis on establishing trust in AI agents trained with reward modelling (Section 6). I would summarize this as: 'Even if we have a perfect solution to the specification problem, it would still be irresponsible to deploy powerful AI systems without having *justified confidence* that we'd solved it.' I believe this, which roughly corresponds to the "Assurance" component in the decomposition of Ortega et al. [228], is the most technically challenging aspect of AI x-safety. I don't expect us to have sufficient theoretical or experimental evidence of x-safety for highly advanced AI systems any time soon; it may be prudent to significantly delay the deployment of such systems while we attempt to establish trustworthy assurance methodologies. The cost of doing so could be large, however – advanced AI systems might lead to dramatically more flourishing (human or otherwise), e.g. by increasing the size of the reachable universe, reducing x-risk from a variety of sources, increasing human health and longevity, enabling rapid scientific and technological progress, etc. The benefits of delaying depend on how likely systems that seem x-safe are to not actually be x-safe, and how much progress we are liable to make on assurance.

1.3.6.3 Hidden Incentives for Auto-Induced Distributional Shift

This work, Krueger et al. [179], is motivated by the problem of instrumental goals. A running example is the goal to influence users of a content recommendation platform in order to make them (e.g.) more engaged or easier to predict. Content recommendation systems can improve their performance by wielding such influence over their users. In our terminology, this creates an **incentive** to manipulate users as a method of increasing performance. However, this incentive need not be "revealed" – a myopic recommender system would not be able to benefit from manipulative actions that only affect users'

future behavior. Our work aims to determine how algorithmic choices influence whether such incentives are revealed. Our main contributions are environments that act as unit tests for revealing non-myopic incentives. And our main results are (potentially) counter-intuitive results where apparently myopic algorithms (such as Q-learning with $\gamma = 0$) nonetheless reveal nonmyopic incentives.

1.4 Alignment Motivation for the Articles Presented

I'll now begin discussing the four articles that make up the bulk of this thesis. I've grouped them into pairs, based on two themes: Generalization and Bayesian Deep Learning. The relevance of these topics to AI Alignment is discussed in Section 1.4.1 and Section 1.4.2, respectively.

1.4.1 Generalization Requirements for Alignment

The first two articles of this thesis deal with generalization. The first contains an empirical study of generalization and memorization in deep learning. The second proposes and studies a novel method for out-of-distribution generalization. Generalization has always been an absolutely central concern of ML. Here, we discuss how generalization relates to AI Alignment in particular (as opposed to more run-of-the-mill AI 'capabilities'), and the unique desiderata and considerations raised by AI Alignment.

1.4.1.1 How *do* Deep Learning systems generalize?

In order for a machine learning-based AI system to learn the right thing from the data the user provides – i.e. for the user and AI to be aligned, they may need some shared understanding of the world in order for the AI to interpret the provided data appropriately. Classic hypothetical alignment failure scenarios often seem to hinge on an AI system misunderstanding a human instruction, e.g. "Make me a sandwich" is interpreted as "Make me into a sandwich" instead of "Prepare me a sandwich". Misunderstanding humans is probably a sufficient condition for AI alignment to fail, although not a necessary one – AI systems could understand what humans mean and still not "care". This makes a

mechanistic understanding of *how* AI systems achieve good performance (e.g. how they generalize well to an i.i.d. test set) important for alignment.

Will AI Systems Learn Human Concepts? A long-standing aspiration of Deep Learning is to learn representations that disentangle factors of variation underlying the data [20, 21]. Implicit in this vision is the assumption that humans do a good job of doing this – i.e. that an AI system that understands the underlying patterns in the data would tend to understand the data in the same way a person would, at least for data that is human-intelligible. Another way of putting this would be to say that human concepts tend to be "natural kinds" [26] – that is, they are intrinsically sensible ways of understanding the world, rather than arbitrary conventions. While there is doubtless *some* truth in this, the existence of adversarial examples [116, 277], and the finding that they correspond to genuine features in the data [154], call this assumption into question. Adversarial examples were discovered by Szegedy et al. [276], and were originally defined as images that look like one thing to humans, but something else to a machine learning model.[4] Adversarial examples have been the subject of thousands of research papers, and yet remain an unsolved problem for machine learning models. Note that the practical significance of adversarial examples as a threat to the reliability of deployed machine learning systems has not been established. While a number of works have focused on adversarial examples as a hypothetical security risk, e.g. demonstrating that real-world adversarial examples can be created by attaching stickers to physical objects [89], Gilmer et al. [107] argue that there is not yet a plausible threat model motivating such concerns. Adversarial examples remain troubling, however, because they indicate that neural networks do not view the world the same way humans do, despite superficially achieving human-level performance on vision tasks [134]. Such differences in how humans and AI systems generalize raises the question: who does it better?

[4] Research on adversarial examples has often focused on ground-truth images subject to norm-limited corruptions chosen "adversarially", i.e. by an optimization process aimed at causing the model to make an incorrect prediction on the corrupted example. For small L_∞-norm perturbations, such corruptions are usually invisible to humans.

Will AI Systems Learn Super-Human Concepts? At a conceptual level, we might expect an AI system with super-human intelligence to have a *better* way of understanding the world, one which might be initially, and even inexorably, alien and unintelligible to us. This would not prevent it from also understanding and being conversant in human concepts. However, we might need to decide whether to trust an AI system that tells us "The correct choice is X, but the reasons for its correctness are beyond your comprehension".[5] A relevant speculative concern is that such an AI system might undergo an **ontological crisis** [69], and realize that the concepts humans use to understand and describe our values are incoherent; this could leave even an intent aligned AI system in a potentially dangerous state of confusion.

Will AI Systems Learn Sub-Human Concepts? On the other hand, there is mounting evidence that current deep learning techniques aren't even equipped to learn the underlying factors of variation. Recently, Schott et al. [257] performed an explicit, systematic evaluation demonstrating that a wide variety of deep learning methods all fail to reliably predict the value of underlying factors of variation for out-of-distribution data, even when those values were provided as supervision during training. It seems that deep learning offers some genuine advantages over other methods, but so far still fails to deliver on its most fantastic promise of learning about underlying factors of variation in a highly generalizable way.

Alignment is easier with human concepts. The distinction between models that learn human concepts and sub-/super-human concepts highlights one aspect of generalization distinctive to alignment. From a capabilities point-of-view, super-human concepts are unproblematic, and even superior to human concepts. From an alignment point-of-view, however, they are deeply problematic, and we would prefer our systems to learn human concepts, at least until we develop justified confidence that the super-human systems are aligned. In terms of x-risk, they are *more* concerning that AI systems with sub-human concepts, since they could outsmart humans. Thus a core question for alignment is whether (what look to us like) generalization failures are actually failures of generalization as opposed to failures of alignment. If a machine learning system can generalize better than a

[5] A similar issue arises when using unrealizable alignment targets (Section 1.3.1).

human in-distribution, but generalizes differently than a human would out-of-distribution, it is not immediately apparent whether this is because humans generalize better out-of-distribution, or if it is because humans simply have different inductive biases. Even if we are prepared to accept that the way humans generalize is correct, this doesn't mean that the AI system won't be better at generalizing in situations where humans' inductive biases fail us.

Summary: Generalizing well and generalizing like a human need not be the same thing. AI systems might conceptualize particular domains, or the world at large, differently from humans. And adversarial examples demonstrate that current AI systems do so. AI concepts could even be *super*-human; e.g. super-human performance on ImageNet was claimed by He et al. [134]. Super-human performance on one benchmark doesn't necessarily translate to super-human understanding, but these results do suggest that super-human understanding could be achieved using concepts quite alien to humans. While this would be a dramatic success from a capabilities point-of-view, it would be quite bad news for x-safety, since we would not be able to understand or trust these systems (without advances in interpretability or assurance more broadly), engendering a significant safety-performance. Understanding how AI systems generalize, and in particular how state-of-the-art deep learning systems generalize, can help inform us about the likelihood of such scenarios. It can also help us understand which techniques are more likely to lead AI systems to generalize as humans would. This provides an alignment motivation for the work in Chapter 2.

1.4.1.2 Out-of-distribution generalization and AI alignment

Given the ability of AI systems to achieve superhuman performance on various tasks without generalizing as humans would on out-of-distribution examples, it is clear that iid performance is not a good measure of the kind of generalization that we care about. The kind of generalization we care about from an alignment point of view is not something that can be easily defined or measured. We care about the more qualitative question of whether an AI system generalizes like a human would. How can we determine whether in AI system does in fact generalize like a human would?

A useful source of information is the behavior of an AI system in entirely novel situations different from those it encountered during training. out-of-distribution generalization is one way of formalizing this idea, by evaluating an AI system on a distribution (or distributions) that is(/are) different from the training distribution. This provides an alignment motivation to study out-of-distribution generalization.

Practically speaking, adversarial examples and other failures of robustness show that good iid performance is no guarantee of good performance out-of-distribution. As an aside, iid performance may not even be a good measure of good behavior *in* distribution, because it could fail to detect extremely rare-but-costly mistakes, and because our performance metric may fail to reflect the true cost of different kinds of mistakes. Theoretically, it is not possible to guarantee good out-of-distribution performance without making assumptions about the out-of-distribution data [68]. In other words, the goal of good generalization is underspecified by the typical iid training/testing regime [66].

Underspecification is not necessarily a problem from a capabilities standpoint: any solution which performs well might be deemed acceptable. However, from an alignment standpoint, we often have preferences about the behavior that are not fully specified by the objective function. Aligning AI systems may require more fully specifying how they should generalize, especially in the absence of shared concepts.

One particular variant of this concern is related to the problem of *inner* alignment. An inner alignment failure is a failure of alignment that occurs not because the objective function is incorrect, but rather because the AI system ends up optimizing a proxy objective [151]. Inner alignment failures could happen for a number of reasons, and some such failures might only be apparent or problematic out-of-distribution.[6]

[6]Inner alignment is closely related to – and can be thought of as a modern take on – the idea of optimization daemons [7]. Taylor [279] present an argument for optimization daemons / inner misalignment being a critical problem for AI x-risk. To summarize/paraphrase: no existing learning algorithm A_{OUTER} seems likely to find a superintelligent AGI design via a direct search; thus A_{OUTER} does find one, it probably did so by instantiating a better learning algorithm A_{INNER} which found the superintelligent AGI. So far, this argument remains mostly speculative, but it could have important consequences if true: it would imply that even models trained with supervised learning or other myopic learning algorithms could end up exhibiting dangerous instrumental goals. Such concerns may seem far fetched at first pass. But imagine training an AI system to (almost) perfectly imitate a human using supervised learning. This would seem to require it learning to do the same sort of approximate planning that a human does. It seems likely that the AI system would learn to do some sort of planning before it learned exactly how the human it is being trained to

Summary: out-of-distribution behavior can reveal more about how an AI system generalizes, helping us to determine if the system is aligned. out-of-distribution generalization also exemplifies the problem of underspecification. In practice, tackling out-of-distribution generalization may require specifying how to generalize, which also seems necessary for alignment.

1.4.2 How could Bayesian (Deep) Learning help with AI Alignment?

The final two articles presented in this thesis are part of a line of work on modeling AI systems' uncertainty in order to guide learning and decision-making towards safer and more aligned outcomes. Specifically, they make progress in the area of **Bayesian Deep Learning**. A primary goal of Bayesian methods to capture uncertainty about which model is best. This is known as **epistemic uncertainty** and can be contrasted with **aleatoric uncertainty**, which is (roughly) inherent randomness.

I see two main benefits to capturing epistemic uncertainty for AI Alignment. First, high epistemic uncertainty about how to make a specific decision indicates that the AI system does not have a good understanding of how to make that decision. In such cases, it may be prudent to fall back on some known-to-be-safe default behavior, such as deferring to human decision-makers. Second, it may about indicate that more information about how to make this specific decision (e.g. human feedback) may be particularly valuable. These two use cases roughly correspond to the problems of **error detection** and **active learning**. These problems could be very relevant for Deep Learning in particular, since Deep Learning systems are notoriously overconfident [124] and data-hungry [182].

From a x-safety perspective, error detection is arguably more important than active learning – after all, a system which recognizes and obeys its own limitations can avoid taking actions its overseer would deem unsafe. However, the bar for error detection as a safety technique may be impossibly high, since even a single error could lead to human extinction. For instance, an AI system which decides whether a piece of code is safe to run could lead to extinction if it approved code that instantiated a misaligned prepotent

imitate does planning (e.g. what sort of goals they have), and thus would be a misaligned planning process, and liable to pursue dangerous instrumental goals.

AI system.

On the other hand, with active learning, the stakes are much lower – intuitively, all that is required is that the epistemic uncertainty provides some meaningful signal about which points are more valuable to query.[7] Furthermore, error detection is merely one strategy for avoiding unacceptable risk-taking. A specification-based approach – i.e. learning what kinds of risk-taking an overseer finds acceptable – might be a promising alternative, and active learning could help with this approach.

Still I believe error detection should have some role to play in AI Alignment; while reliably detecting potential errors is hard, it does seem easier than reliably making the right decisions! An important outstanding question is: how much easier? I've come up with a simple information-theoretic argument (which may or may not be novel) for why it should be significantly easier. First, communicating whether or not an AI knows how to make a decision competently only requires one bit, whereas communicating what the decision should be requires may require many more bits. Moreover, consider a region that is simple to describe (e.g. all inputs of norm greater than 1), where the AI is incompetent and containing many different situations where different behaviors are required. It is much simpler to describe this region as unsafe than it is to specify how to behave in each area within the region. Exploring this line of argument could be an interesting direction for future work.

1.4.2.1 A summary of my works on Bayesian methods, and my motivations

The main idea of Bayesian machine learning is to compute the posterior distribution over model parameters, given the observed training data. Bayesian Deep Learning requires approximating this posterior. When combined, the two articles presented in the next two chapters of this thesis yield a method that is capable – in principle – of approximating the true posterior of a DNN arbitrarily well. Bayesian Hypernetworks (Section 6) show how to use normalizing flows as approximate variational posteriors for Bayesian Deep Learning. Neural Autoregressive Flows (Section 8) shows how to

[7] It is worth noting that even active learning remains a challenging problem in machine learning – uniform random query selection remains a strong baseline [199, 313].

use DNNs to construct normalizing flows expressive enough to approximate arbitrary probability distributions arbitrarily well.

My research interest in Bayesian Deep Learning sprung out of my earlier work on Active Reinforcement Learning (Active RL) [178], which is not included in this thesis. Still, I will provide a summary, as it helps explain the Alignment motivation for the two articles which are included. The motivation for active RL is can be summarized as:

1. Misspecified reward functions may lead to dangerous perverse instantiations
2. Humans may struggle to accurately specify a reward function, which dictates the reward for *any* possible situation (i.e. state) that a reinforcement learning agent might encounter.
3. Assigning rewards on demand for states that an agent actually encounters could be significantly easier.
4. However, the human labor involved is doing so is costly.
5. Thus an agent making use of such on-demand reward signals must have a way of deciding when it is worth paying this cost.

In Krueger et al. [178], we formalized this setting as a variant of standard RL, and propose and evaluate Bayesian RL methods on several illustrative tabular environments. Our Bayesian methods decided when to query the (simulated) human based on an estimate of how valuable the resulting reward signal might be; these query decisions resemble the query decisions in active learning and lead us to call the problem setting "Active RL".

A next step was to scale this to continuous state-spaces, using DNNs for function approximation. This is what lead me to explore methods for Bayesian Deep Learning. At this point, the leading method for Bayesian Deep Learning was MCDropout [98], which interprets dropout noise as generating a variational posterior. However, in studying MCDropout, I realized that existing variational posteriors were quite limited in their expressive power (e.g. the fixed noise of dropout, "mean-field" posteriors with no dependence between parameters), and that it would be possible to use a deep generative model

as a more flexible variational posterior, so long as it also had tractable likelihood. This is what lead me to work on Bayesian Hypernetworks.

The posterior of a neural network is believed to be multimodal, since different parameter values produce the same function [237]. Although we showed that Bayesian Hypernetworks could, in principle, learn to represent multimodal posterior distributions, we also noted that in practice the normalizing flow models that we used in that work struggled to learn multimodal distributions. This was the motivation for our work on Neural Autoregressive Flows, which included developing more flexible normalizing flows more suited for modeling multimodal distributions.

Another work in Bayesian Deep Learning that I made minor contributions to – which is not featured in this thesis – is Lacoste et al. [181].

CHAPTER 2

DISCUSSION OF "A CLOSER LOOK AT MEMORIZATION IN DEEP NETWORKS"

Author List

Devansh Arpit*
Montréal Institute for Learning Algorithms, Canada;
Université de Montréal, Canada

Stanisław Jastrzębski*
Jagiellonian University, Krakow, Poland

Nicolas Ballas*
Montréal Institute for Learning Algorithms, Canada;
Université de Montréal, Canada

David Krueger*
Montréal Institute for Learning Algorithms, Canada;
Université de Montréal, Canada

Emmanuel Bengio
McGill University, Canada

Maxinder S. Kanwal
University of California, Berkeley, USA

Tegan Maharaj
Montréal Institute for Learning Algorithms, Canada;
Polytechnique Montréal, Canada

Asja Fischer
University of Bonn, Bonn, Germany

Aaron Courville
Montréal Institute for Learning Algorithms, Canada;
Université de Montréal, Canada;
CIFAR Fellow

Yoshua Bengio
Montréal Institute for Learning Algorithms, Canada;
Université de Montréal, Canada;
CIFAR Senior Fellow

Simon Lacoste-Julien
Montréal Institute for Learning Algorithms, Canada;
Université de Montréal, Canada

* Equal contributors

2.1 My Contributions

1. I helped to organize initial meetings and manage the project.

2. I conceived the original workshop paper (for which I was the sole first author); the workshop version was expanded substantially for the conference submission.

3. I helped define the organizing themes (e.g. highlighting differences between learning behavior on real/random data, the idea that simple patterns are learned first, often precluding memorization).

4. I devised and ran the experiments in sections 3.1 and 3.3.

5. I did a plurality of the writing, and a lot of editing.

2.2 Background

It is not well understood why Deep Learning methods are able to generalize so well in practice. Statistical learning theory is the main framework which has been developed to explain generalization in machine learning. However, the tools of statistical learning theory have so far failed to explain the success of deep learning.

The basic picture painted by statistical learning theory can be summarized as a balancing act between overfitting and underfitting. Simply put, underfitting refers to using a model that is too simple to capture the underlying patterns in the data, whereas overfitting refers to using a model that is so flexible that it latches onto peculiarities in the training data which do not generalize.

Statistical learning theory views learning as selecting a hypothesis h from some hypothesis class $\mathscr{H}$, typically via the principle of Empirical Risk Minimization (ERM) – that is, choosing whichever model fits the data best. ERM is equivalent to Maximum Likelihood Estimation (MLE) when likelihood is used as a measure of performance. The choice of $\mathscr{H}$ is considered critical, since a hypothesis class that is too small may lead to underfitting, while one that is too large/expressive may lead to overfitting.

A few tacit assumptions of note here are:

- **Empirical Risk Minimization (ERM):** we select the hypothesis that minimizes the training loss.
- **Limiting capacity as key to avoiding overfitting:** an overly flexible hypothesis space will lead to overfitting.
- **Data-independent notions of capacity:** Capacity doesn't depend on the data or learning algorithm.

In 2016, Zhang et al. [316] published a work showing that neural networks can fit random labelings of datasets such as ImageNet with roughly 100% accuracy. This simple finding demonstrated that traditional statistical learning theory could not offer a good explanation for Deep Learning's ability to generalize, calling into question the assumptions listed above. Their work was awarded a best paper award at ICLR 2017, and

seems to have initiated – or at least greatly contributed to – a large research effort effort to provide theoretical explanations for deep learning generalization.

While Zhang et al. [316] emphasized the ability of Deep Learning to memorize random examples, their work left several key questions unanswered:

1. What exactly is memorization, and does it play a role in neural network learning?
2. What is the role of early stopping?
3. How can we reconcile the 'paradoxical' ability of Deep Learning to both generalize and memorize?

Our work helped answer all of these questions; our explanation can be summarized as follows:

1. Deep Learning methods learn simple patterns first, and memorize later.
2. Thus early stopping can help avoid memorization.
3. Early stopping isn't as helpful when there are no incorrectly labeled examples to memorize, but can improve performance when there are.

2.3 Contributions of the work

In this work, we introduced a number of novel approaches to understanding deep networks learning behavior, which have been expanded on in other works. We used these analysis tools to demonstrate striking differences in learning behavior on real vs. random data. Our operationalization of memorization as "behavior on random data" was already implicit in Zhang et al. [316], and since our work, this notion of memorization has been refined. Feldman and Zhang [90] say an example is memorized when it is learned iff it is included in the training data. This (more or less) generalizes the previous notion, since randomly labeled examples will not be predicted correctly (above chance levels) unless they are included in the training set. Broadly speaking, this work played a large role in establishing the sub-field of "empirical theory" in deep learning.

CHAPTER 3

A CLOSER LOOK AT MEMORIZATION IN DEEP NETWORKS

ABSTRACT

We examine the role of memorization in deep learning, drawing connections to capacity, generalization, and adversarial robustness. While deep networks are capable of memorizing noise data, our results suggest that they tend to prioritize learning simple patterns first. In our experiments, we expose qualitative differences in gradient-based optimization of deep neural networks (DNNs) on noise vs. real data. We also demonstrate that for appropriately tuned explicit regularization (e.g., dropout) we can degrade DNN training performance on noise datasets without compromising generalization on real data. Our analysis suggests that the notions of effective capacity which are dataset independent are unlikely to explain the generalization performance of deep networks when trained with gradient based methods because training data itself plays an important role in determining the degree of memorization.

3.1 Introduction

The traditional view of generalization holds that a model with sufficient capacity (e.g. more parameters than training examples) will be able to "memorize" each example, overfitting the training set and yielding poor generalization to validation and test sets [118]. Yet deep neural networks (DNNs) often achieve excellent generalization performance with massively over-parameterized models. This phenomenon is not well-understood.

From a representation learning perspective, the generalization capabilities of DNNs are believed to stem from their incorporation of good generic priors (see, e.g., Bengio et al. [21]). Lin and Tegmark [197] further suggest that the priors of deep learning are well suited to the physical world. But while the priors of deep learning may help explain why DNNs learn to efficiently represent complex real-world functions, they are not restrictive enough to rule out memorization.

On the contrary, deep nets are known to be universal approximators, capable of representing arbitrarily complex functions given sufficient capacity [65, 146]. Furthermore, recent work has shown that the expressiveness of DNNs grows exponentially with depth [216, 236]. These works, however, only examine the *representational capacity*, that is, the set of hypotheses a model is capable of expressing via some value of its parameters.

Because DNN optimization is not well-understood, it is unclear which of these hypotheses can actually be reached by gradient-based training [37]. In this sense, optimization and generalization are entwined in DNNs. To account for this, we formalize a notion of the *effective capacity (EC)* of a learning algorithm $\mathscr{A}$ (defined by specifying both the model *and the training procedure*, e.g.,"train the LeNet architecture [187] for 100 epochs using stochastic gradient descent (SGD) with a learning rate of 0.01") as the set of hypotheses which can be reached by applying that learning algorithm on *some* dataset. Formally, using set-builder notation:

$$EC(\mathscr{A}) = \{h \mid \exists \mathscr{D} \text{ such that } h \in \mathscr{A}(\mathscr{D})\} \ ,$$

where $\mathscr{A}(\mathscr{D})$ represents the set of hypotheses that is reachable by $\mathscr{A}$ on a dataset $\mathscr{D}$[1].

One might suspect that DNNs effective capacity is sufficiently limited by gradient-based training and early stopping to resolve the apparent paradox between DNNs' excellent generalization and their high representational capacity. However, the experiments of Zhang et al. [317] suggest that this is not the case. They demonstrate that DNNs are able to fit pure noise without even needing substantially longer training time. Thus even the *effective* capacity of DNNs may be too large, from the point of view of traditional learning theory.

By demonstrating the ability of DNNs to "memorize" random noise, Zhang et al. [317] also raise the question whether deep networks use similar memorization tactics on real datasets. Intuitively, a brute-force memorization approach to fitting data does not capitalize on patterns shared between training examples or features; the *content* of what is memorized is irrelevant. A paradigmatic example of a memorization algorithm is k-

[1]Since $\mathscr{A}$ can be stochastic, $\mathscr{A}(\mathscr{D})$ is a set.

nearest neighbors [91]. Like Zhang et al. [317], we do not formally define memorization; rather, we investigate this intuitive notion of memorization by training DNNs to fit random data.

Main contributions We operationalize the definition of "memorization" as *the behavior exhibited by DNNs trained on noise*, and conduct a series of experiments that contrast the learning dynamics of DNNs on real vs. noise data. Thus, our analysis builds on the work of Zhang et al. [317] and further investigates the role of memorization in DNNs.

Our findings are summarized as follows:

1. There are qualitative differences in DNN optimization behavior on real data vs. noise. In other words, DNNs do not just memorize real data (subsection 3.3).

2. DNNs learn simple patterns first, before memorizing (subsection 3.4). In other words, DNN optimization is *content-aware*, taking advantage of patterns shared by multiple training examples.

3. Regularization techniques can differentially hinder memorization in DNNs while preserving their ability to learn about real data (subsection 3.5).

3.2 Experiment details

We perform experiments on MNIST [187] and CIFAR10 [173] datasets. We investigate two classes of models: 2-layer multi-layer perceptrons (MLPs) with rectifier linear units (ReLUs) on MNIST and convolutional neural networks (CNNs) on CIFAR10. If not stated otherwise, the MLPs have 4096 hidden units per layer and are trained for 1000 epochs with SGD and learning rate 0.01. The CNNs are a small Alexnet-style CNN[2] (as in Zhang et al. [317]), and are trained using SGD with momentum=0.9 and learning rate of 0.01, scheduled to drop by half every 15 epochs.

[2]Input $\rightarrow$ Crop(2,2) $\rightarrow$ Conv(200,5,5) $\rightarrow$ BN $\rightarrow$ ReLU $\rightarrow$ MaxPooling(3,3) $\rightarrow$ Conv(200,5,5) $\rightarrow$ BN$\rightarrow$ ReLU$\rightarrow$ MaxPooling(3,3) $\rightarrow$ Dense(384) $\rightarrow$ BN $\rightarrow$ ReLU $\rightarrow$ Dense(192) $\rightarrow$ BN $\rightarrow$ ReLU $\rightarrow$ Dense(#classes) $\rightarrow$ Softmax. Here Crop(. , .) crops height and width from both sides with respective values.

Following Zhang et al. [317], in many of our experiments we replace either (some portion of) the labels (with random labels), or the inputs (with i.i.d. Gaussian noise matching the real dataset's mean and variance) for some fraction of the training set. We use *randX* and *randY* to denote datasets with (100%, unless specified) noisy inputs and labels (respectively).

3.3 Qualitative differences of DNNs trained on random vs. real data

Zhang et al. [317] empirically demonstrated that DNNs are capable of fitting random data, which implicitly necessitates some high degree of memorization. In this subsection, we investigate whether DNNs employ similar memorization strategy when trained on real data. In particular, our experiments highlight some qualitative differences between DNNs trained on real data vs. random data, supporting the fact that DNNs do not use brute-force memorization to fit real datasets.

3.3.1 Easy examples as evidence of patterns in real data

A brute-force memorization approach to fitting data should apply equally well to different training examples. However, if a network is learning based on patterns in the data, some examples may fit these patterns better than others. We show that such "easy examples" (as well as correspondingly "hard examples") are common in real, but not in random, datasets. Specifically, for each setting (real data, randX, randY), we train an MLP for a single epoch starting from 100 different random initializations and shufflings of the data. We find that, for real data, many examples are consistently classified (in)correctly after a single epoch, suggesting that different examples are significantly easier or harder in this sense. For noise data, the difference between examples is much less, indicating that these examples are fit (more) independently. Results are presented in Figure 3.3.1.

For randX, apparent differences in difficulty are well modeled as random Binomial noise. For randY, this is not the case, indicating some use of shared patterns. Visualizing first-level features learned by a CNN supports this hypothesis (Figure 3.3.1).

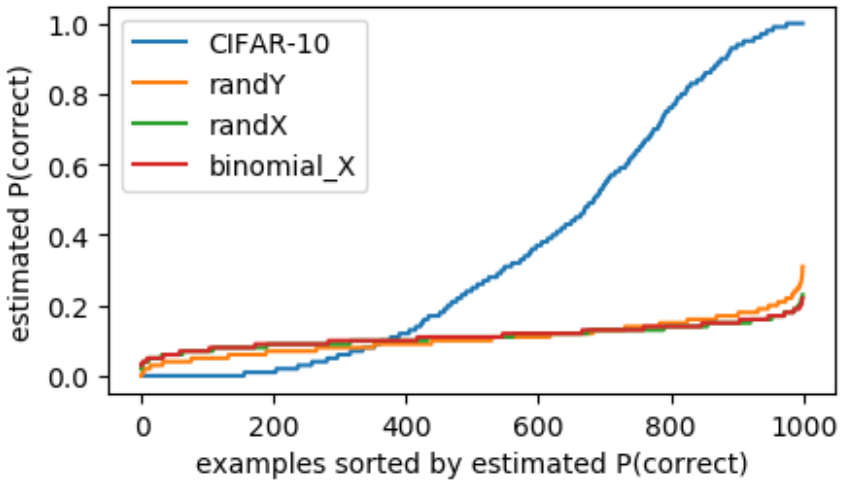


Figure 3.1: Average (over 100 experiments) misclassification rate for each of 1000 examples after one epoch of training. This measure of an example's difficulty is much more variable in real data. We conjecture this is because the easier examples are explained by some simple patterns, which are reliably learned within the first epoch of training. We include 1000 points samples from a binomial distribution with $n = 100$ and p equal to the average estimated P(correct) for randX, and note that this curve closely resembles the randX curve, suggesting that random inputs are all equally difficult.

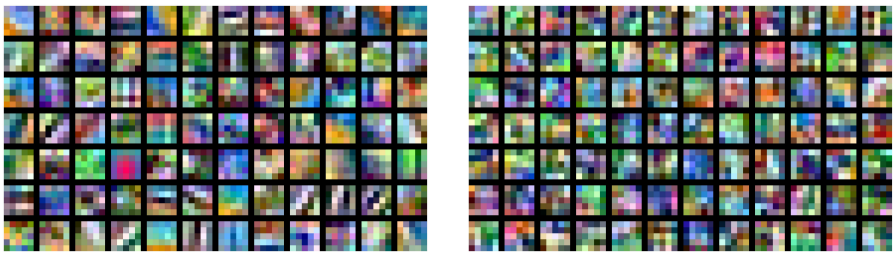


Figure 3.2: Filters from first layer of network trained on CIFAR10 (left) and randY (right).

3.3.2 Loss-sensitivity in real vs. random data

To further investigate the difference between real and fully random inputs, we propose a proxy measure of memorization via gradients. Since we cannot measure quantitatively how much each training sample $\mathbf{x}$ is memorized, we instead measure the effect of each sample on the average loss. That is, we measure the norm of the loss gradient with respect to a previous example $\mathbf{x}$ after t SGD updates. Let $\mathscr{L}_t$ be the loss after t updates; then the

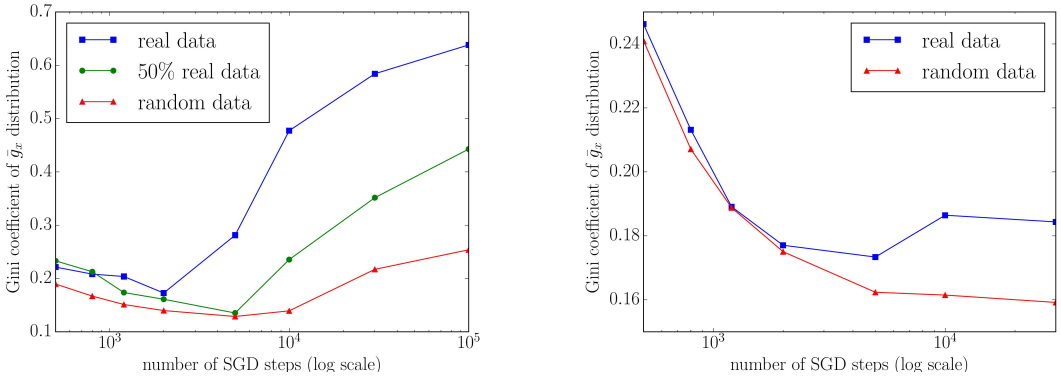


Figure 3.3: Plots of the Gini coefficient of $\bar{g}_{\mathbf{x}}$ over examples $\mathbf{x}$ (see subsection 3.3.2) as training progresses, for a 1000-example real dataset (14x14 MNIST) versus random data. On the left, Y is the normal class label; on the right, there are as many classes as examples, the network has to learn to map each example to a unique class.

sensitivity measure is given by

$$g_{\mathbf{x}}^t = \|\partial \mathscr{L}_t / \partial \mathbf{x}\|_1 \quad .$$

The parameter update from training on $\mathbf{x}$ influences all future $\mathscr{L}_t$ indirectly by changing the subsequent updates on different training examples. We denote the average over $g_{\mathbf{x}}^t$ after T steps as $\bar{g}_{\mathbf{x}}$, and refer to it as *loss-sensitivity*. Note that we only report ℓ^1-norm results, but that results stay very similar using ℓ^2-norm and infinity norm.

We compute $g_{\mathbf{x}}^t$ by unrolling t SGD steps and applying backpropagation over the unrolled computation graph, as done by Maclaurin et al. [208]. Unlike Maclaurin et al. [208], we only use this procedure to compute $g_{\mathbf{x}}^t$, and do not modify the training procedure in any way.

We find that for real data, only a subset of the training set has high $\bar{g}_{\mathbf{x}}$, while for random data, $\bar{g}_{\mathbf{x}}$ is high for virtually all examples. We also find a different behavior when *each example* is given a unique class; in this scenario, the network has to learn to identify each example uniquely, yet still behaves differently when given real data than when given random data as input.

We visualize (Figure 3.3) the spread of $\bar{g}_{\mathbf{x}}$ as training progresses by computing the

Gini coefficient over $\mathbf{x}$'s. The Gini coefficient [108] is a measure of the inequality among values of a frequency distribution; a coefficient of 0 means exact equality (i.e., all values are the same), while a coefficient of 1 means maximal inequality among values. We observe that, when trained on real data, the network has a high $\bar{g}_{\mathbf{x}}$ for a few examples, while on random data the network is sensitive to most examples. The difference between the random data scenario, where we know the neural network needs to do memorization, and the real data scenario, where we're trying to understand what happens, leads us to believe that this measure is indeed sensitive to memorization. Additionally, these results suggest that when being trained on real data, the neural network probably does not memorize, or at least not in the same manner it needs to for random data.

In addition to the different behaviors for real and random data described above, we also consider a class specific loss-sensitivity: $\bar{g}_{i,j} = \mathbb{E}_{(x,y)}{}^1\!/_T \sum_t^T |\partial \mathscr{L}_t(y=i)/\partial x_{y=j}|$, where $\mathscr{L}_t(y=i)$ is the term in the crossentropy sum corresponding to class i. We observe that the loss-sensitivity w.r.t. class i for training examples of class j is higher when $i=j$, but more spread out for real data (see Figure 3.4). An interpretation of this is that for real data there are more interesting cross-category patterns that can be learned than for random data.

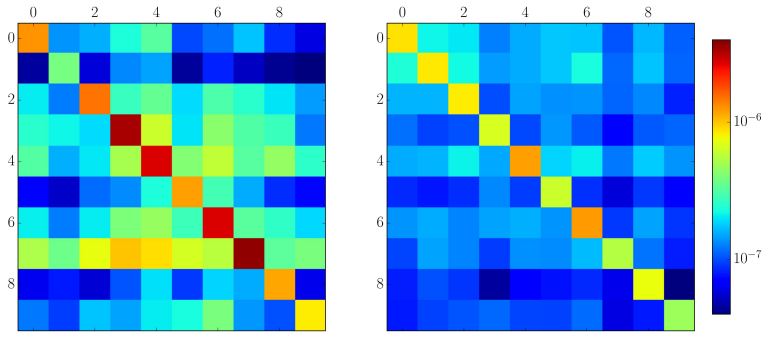


Figure 3.4: Plots of per-class g_x (see previous figure; log scale), a cell i,j represents the average $|\partial \mathscr{L}(y=i)/\partial x_{y=j}|$, i.e. the loss-sensitivity of examples of class i w.r.t. training examples of class j. Left is real data, right is random data.

Figure 3.3 and 3.4 were obtained by training a fully-connected network with 2 layers of 16 units on 1000 downscaled 14×14 MNIST digits using SGD.

3.3.3 Capacity and effective capacity

In this subsection, we investigate the impact of capacity and effective capacity on learning of datasets having different amounts of random input data or random labels.

3.3.3.0.1 Effects of capacity and dataset size on validation performances In a first experiment, we study how overall model capacity impacts the validation performances for datasets with different amounts of noise. On MNIST, we found that the optimal validation performance requires a higher capacity model in the presence of noise examples (see Figure 3.5). This trend was consistent for noise inputs on CIFAR10, but we did not notice any relationship between capacity and validation performance on random *labels* on CIFAR10.

This result contradicts the intuitions of traditional learning theory, which suggest that capacity should be restricted, in order to enforce the learning of (only) the most regular patterns. Given that DNNs can perfectly fit the training set in any case, we hypothesize that that higher capacity allows the network to fit the noise examples in a way that does not interfere with learning the real data. In contrast, if we were simply to *remove* noise examples, yielding a smaller (clean) dataset, a *lower* capacity model would be able to achieve optimal performance.

3.3.3.0.2 Effects of capacity and dataset size on training time Our next experiment measures time-to-convergence, i.e. how many epochs it takes to reach 100% training accuracy. Reducing the capacity or increasing the size of the dataset slows down training as well for real as for noise data[3]. However, the effect is more severe for datasets containing noise, as our experiments in this subsection show (see Figure 3.6).

Effective capacity of a DNN can be increased by increasing the representational capacity (e.g. adding more hidden units) or training for longer. Thus, increasing the number of hidden units decreases the number of training iterations needed to fit the data, up to some limit. We observe *stronger* diminishing returns from increasing representational

[3]Regularization can also increase time-to-convergence; see subsection 3.5.

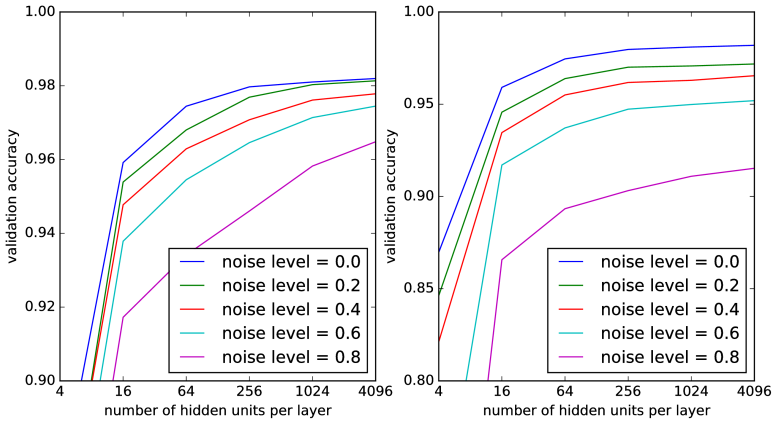


Figure 3.5: Performance as a function of capacity in 2-layer MLPs trained on (noisy versions of) MNIST. For real data, performance is already very close to maximal with 4096 hidden units, but when there is noise in the dataset, higher capacity is needed.

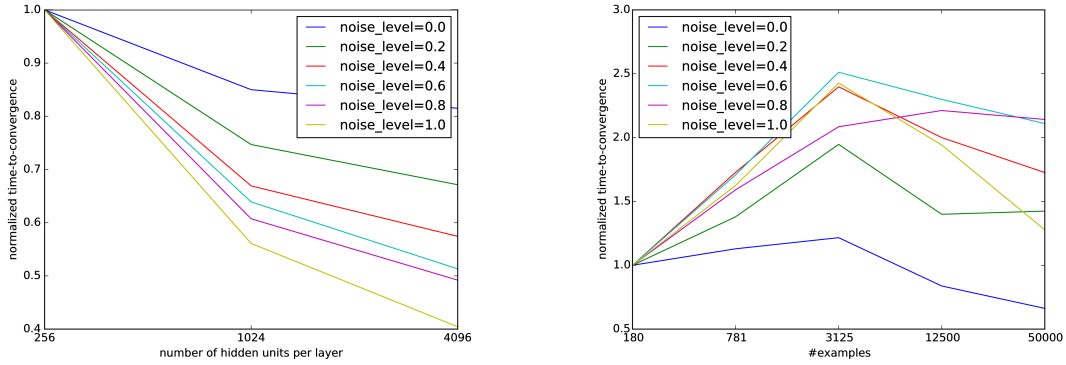


Figure 3.6: Time to convergence as a function of capacity with dataset size fixed to 50000 (left), or dataset size with capacity fixed to 4096 units (right). "Noise level" denotes to the proportion of training points whose inputs are replaced by Gaussian noise. Because of the patterns underlying real data, having more capacity/data does not decrease/increase training time as much as it does for noise data.

capacity for real data, indicating that this limit is lower, and a smaller representational capacity is sufficient, for real datasets.

Increasing the number of examples (keeping representational capacity fixed) also increases the time needed to memorize the training set. In the limit, the representational capacity is simply insufficient, and memorization is not feasible. On the other hand, when

the relationship between inputs and outputs is meaningful, new examples simply give more (possibly redundant) clues as to what the input $\rightarrow$ output mapping is. Thus, in the limit, an idealized learner should be able to predict unseen examples perfectly, absent noise. Our experiments demonstrate that time-to-convergence is not only longer on noise data (as noted by Zhang et al. [317]), but also, *increases* substantially as a function of dataset size, relative to real data. Following the reasoning above, this suggests that our networks are learning to extract patterns in the data, rather than memorizing.

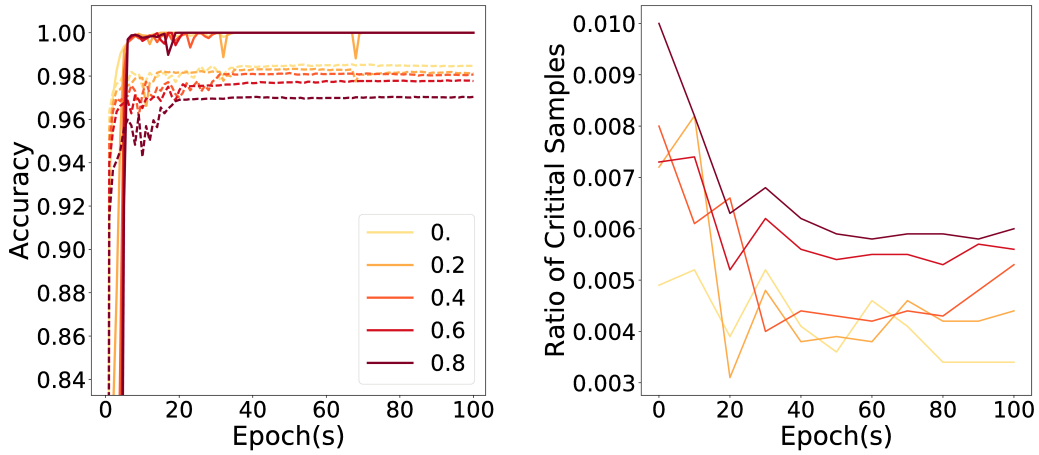
3.4 DNNs learn patterns first

This subsection aims at studying how the complexity of the hypotheses learned by DNNs evolve during training for real data vs. noise data. To achieve this goal, we build on the intuition that the number of different decision regions into which an input space is partitioned reflects the complexity of the learned hypothesis [268]. This notion is similar in spirit to the degree to which a function can scatter random labels: a higher density of decision boundaries in the data space allows more samples to be scattered.

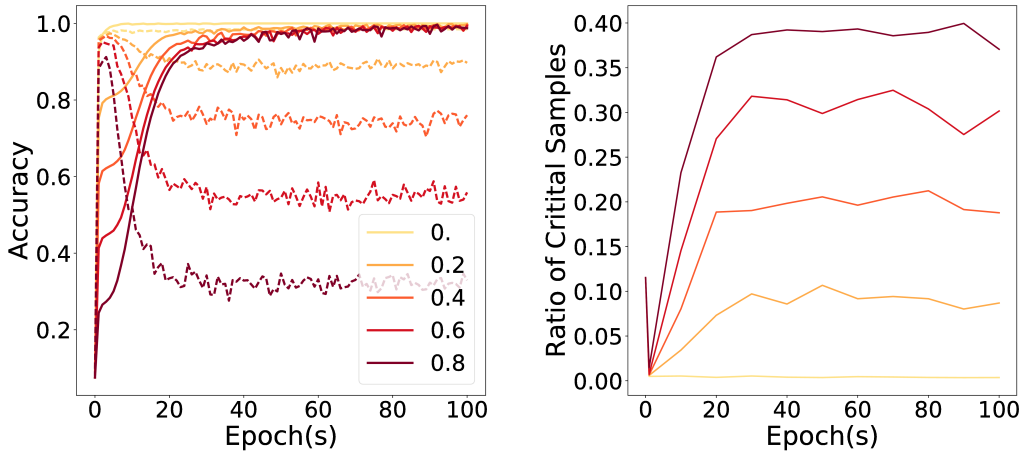
Therefore, we estimate the complexity by measuring how densely points on the data manifold are present around the model's decision boundaries. Intuitively, if we were to randomly sample points from the data distribution, a smaller fraction of points in the proximity of a decision boundary suggests that the learned hypothesis is simpler.

3.4.1 Critical Sample Ratio (CSR)

Here we introduce the notion of a *critical sample*, which we use to estimate the density of decision boundaries as discussed above. Critical samples are a subset of a dataset such that for each such sample $\mathbf{x}$, there exists at least one adversarial example $\hat{\mathbf{x}}$ in the proximity of $\mathbf{x}$. Specifically, consider a classification network's output vector $f(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_k(\mathbf{x})) \in \mathbb{R}^k$ for a given input sample $\mathbf{x} \in \mathbb{R}^n$ from the data manifold.



(a) Noise added on classification inputs.



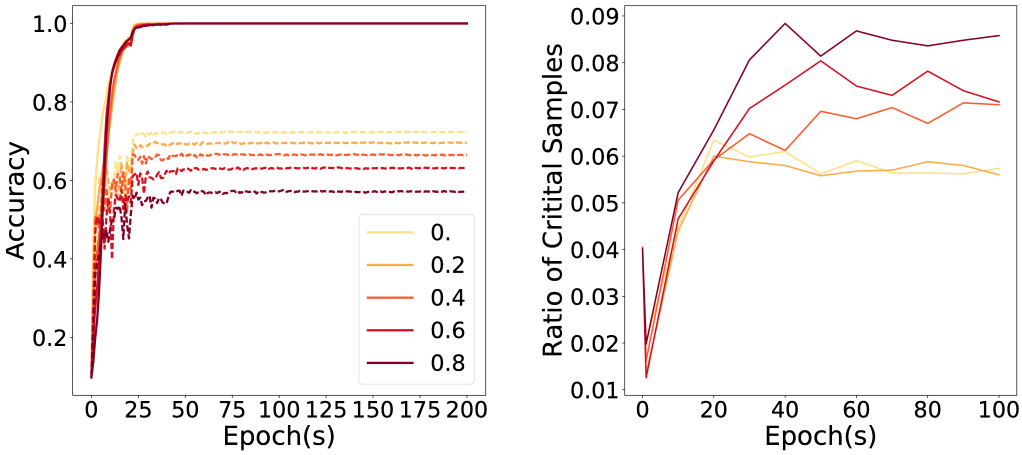
(b) Noise added on classification labels.

Figure 3.7: Accuracy (left in each pair, solid is train, dotted is validation) and Critical sample ratios (right in each pair) for MNIST.

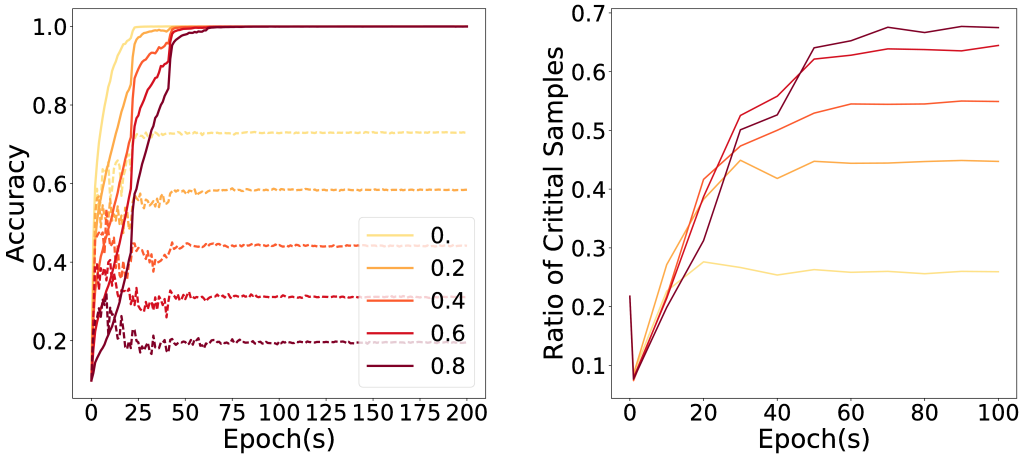
Formally we call a dataset sample $\mathbf{x}$ a *critical sample* if there exists a point $\hat{\mathbf{x}}$ such that,

$$\arg\max_i f_i(\mathbf{x}) \neq \arg\max_j f_j(\hat{\mathbf{x}}) \tag{3.1}$$

$$\text{s.t. } \|\mathbf{x} - \hat{\mathbf{x}}\|_\infty \leq r$$



(a) Noise added on classification inputs.



(b) Noise added on classification labels.

Figure 3.8: Accuracy (left in each pair, solid is train, dotted is validation) and Critical sample ratios (right in each pair) for CIFAR10.

where r is a fixed box size. As in recent work on adversarial examples [180] the above definition depends only on the predicted label $\arg\max_i f_i(\mathbf{x})$ of $\mathbf{x}$, and not the true label (as in earlier work on adversarial examples, such as Goodfellow et al. [115], Szegedy et al. [275]).

Following the above argument relating complexity to decision boundaries, a higher number of critical samples indicates a more complex hypothesis. Thus, we measure

complexity as the *critical sample ratio (CSR)*, that is, the fraction of data-points in a set $|\mathscr{D}|$ for which we can find a critical sample: $\frac{\#\text{critical samples}}{|\mathscr{D}|}$.

To identify whether a given data point $\mathbf{x}$ is a critical samples, we search for an adversarial sample $\hat{\mathbf{x}}$ within a box of radius r. To perform this search, we propose using Langevin dynamics applied to the fast gradient sign method (FGSM, Goodfellow et al. [115]) as shown in algorithm 1[4]. We refer to this method as Langevin adversarial sample search (LASS). While the FGSM search algorithm can get stuck at a points with zero gradient, LASS explores the box more thoroughly. Specifically, a problem with first order gradient search methods (like FGSM) is that there might exist training points where the gradient is 0, but with a large 2^{nd} derivative corresponding to a large change in prediction in the neighborhood. The noise added by the LASS algorithm during the search enables escaping from such points.

Algorithm 1 Langevin Adversarial Sample Search (LASS)

Require: $\mathbf{x} \in \mathbb{R}^n$, α, β, r, noise process η
Ensure: $\hat{\mathbf{x}}$

converged = FALSE
$\tilde{\mathbf{x}} \leftarrow \mathbf{x}$; $\hat{\mathbf{x}} \leftarrow \emptyset$
while not converged or max iter reached **do**
  $\Delta = \alpha \cdot \text{sign}(\frac{\partial f_k(\mathbf{x})}{\partial \mathbf{x}}) + \beta \cdot \eta$
  $\tilde{\mathbf{x}} \leftarrow \tilde{\mathbf{x}} + \Delta$
  for $i \in [n]$ **do**
    $\tilde{\mathbf{x}}_i \leftarrow \begin{cases} \mathbf{x}_i + r \cdot \text{sign}(\tilde{\mathbf{x}}_i - \mathbf{x}^i) & if |\tilde{\mathbf{x}}_i - \mathbf{x}_i| > r \\ \tilde{\mathbf{x}}_i & otherwise \end{cases}$
  end for
  if $\arg\max_i f(\mathbf{x}) \neq \arg\max_i f(\tilde{\mathbf{x}})$ **then**
    converged = TRUE
    $\hat{\mathbf{x}} \leftarrow \tilde{\mathbf{x}}$
  end if
end while

[4]In our experiments, we set $\alpha = 0.25$, $\beta = 0.2$ and η is samples from standard normal distribution.

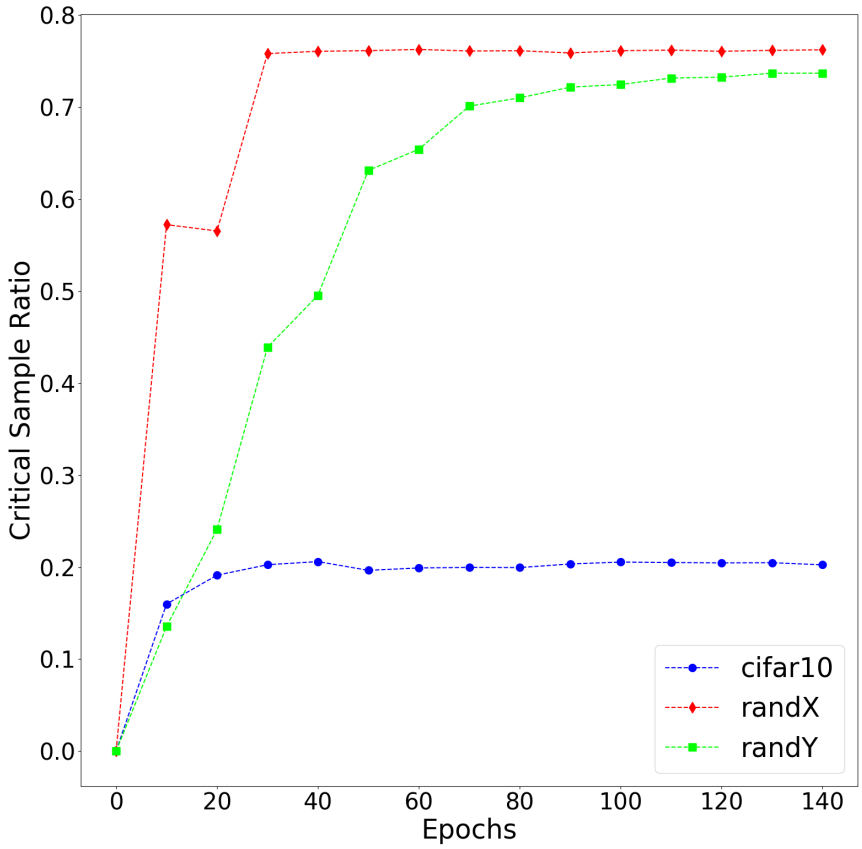


Figure 3.9: Critical sample ratio throughout training on CIFAR-10, random input (randX), and random label (randY) datasets.

3.4.2 Critical samples throughout training

We now show that the number of critical samples is much higher for a deep network (specifically, a CNN) trained on noise data compared with real data. To do so, we measure the number of critical samples in the validation set[5], throughout training[6]. Results are shown in Figure 3.9. A higher number of critical samples for models trained on noise data compared with those trained on real data suggests that the learned decision surface is

[5]We also measure the number of critical samples in the training sets. Since we train our models using log loss, training points are pushed away from the decision boundary even after the network learns to classify them correctly. This leads to an initial rise and then fall of the number of critical samples in the training sets.

[6]We use a box size of 0.3, which is small enough in a 0-255 pixel scale to be unnoticeable by a human evaluator. Different values for r were tested but did not change results qualitatively and lead to the same conclusions

more complex for noise data (randX and randY). We also observe that the CSR increases gradually with increasing number of epochs and then stabilizes. This suggests that the networks learn gradually more complex hypotheses during training for all three datasets.

In our next experiment, we evaluate the performance and critical sample ratio of datasets with 20% to 80% of the training data replaced with either input or label noise. Results for MNIST and CIFAR-10 are shown in Figures 3.7 and 3.8, respectively. For both randX and randY datasets, the CSR is higher for noisier datasets, reflecting the higher level of complexity of the learned prediction function. The final and maximum validation accuracies are also both lower for noisier datasets, indicating that the noise examples interfere somewhat with the networks ability to learn about the real data.

More significantly, for randY datasets (Figures 3.7(b) and 3.8(b)), the network achieves maximum accuracy on the validation set before achieving high accuracy on the training set. Thus the model first learns the simple and general patterns of the real data before fitting the noise (which results in decreasing validation accuracy). Furthermore, as the model moves from fitting real data to fitting noise, the CSR greatly increases, indicating the need for more complex hypotheses to explain the noise. Combining this result with our results from subsection 3.3.1, we conclude that real data examples are easier to fit than noise.

3.5 Effect of regularization on learning

Here we demonstrate the ability of regularization to degrade training performance on data with random labels, while maintaining generalization performance on real data. Zhang et al. [317] argue that explicit regularizations are not the main explanation of good generalization performance, rather SGD based optimization is largely responsible for it. Our findings extend their claim and indicate that explicit regularizations can substantially limit the speed of memorization of noise data without significantly impacting learning on real data.

We compare the performance of CNNs trained on CIFAR-10 and randY with the following regularizers: dropout (with dropout rates in range 0-0.9), input dropout (range

0-0.9), input Gaussian noise (with standard deviation in range 0-5), hidden Gaussian noise (range 0-0.3), weight decay (range 0-1) and additionally dropout with adversarial training (with weighting factor in range 0.2-0.7 and dropout in rate range 0.03-0.5).[7] We train a separate model for every combination of dataset, regularization technique, and regularization parameter.

The results are summarized in Figure 3.5. For each combination of dataset and regularization technique, the final training accuracy on randY (x-axis) is plotted against the best validation accuracy on CIFAR-10 from amongst the models trained with different regularization parameters (y-axis). Flat curves indicate that the corresponding regularization technique can reduce memorization when applied on random labeling, while resulting in the same validation accuracy on the clean validation set. Our results show that different regularizers target memorization behavior to different extent – dropout being the most effective. We find that dropout, especially coupled with adversarial training, is best at hindering memorization without reducing the model's ability to learn. Figure 3.5 additionally shows this effect for selected experiments (i.e. selected hyperparameter values) in terms of train loss.

3.6 Related work

Our work builds on the experiments and challenges the interpretations of Zhang et al. [317]. We make heavy use of their methodology of studying DNN training in the context of noise datasets. Zhang et al. [317] show that DNNs can perfectly fit noise and thus that their generalization ability cannot be explained through traditional statistical learning theory (e.g., see [15, 297]). We agree with this finding, but show in addition that the degree of memorization and generalization in DNNs depends not only on the architecture and training procedure (including explicit regularizations), *but also on the training data itself*[8].

Another direction we investigate is the relationship between regularization and mem-

[7] We perform adversarial training using critical samples found by LASS algorithm with default parameters.

[8] We conclude the latter part based on experimental findings in subsections 3.3 and 3.4.2

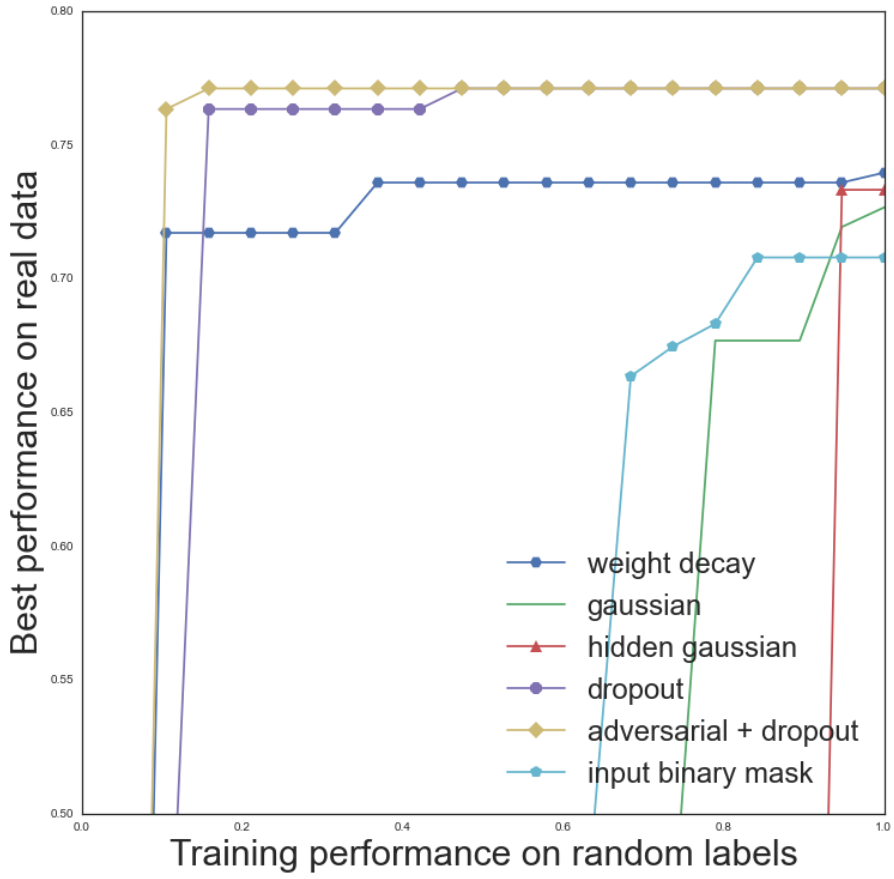


Figure 3.10: Effect of different regularizers on train accuracy (on noise dataset) vs. validation accuracy (on real dataset). Flatter curves indicate that memorization (on noise) can be capped without sacrificing generalization (on real data).

orization. Zhang et al. [317] argue that explicit and implicit regularizers (including SGD) might not explain or limit shattering of random data. In this work we show that regularizers (especially dropout) *do* control the *speed* at which DNNs memorize. This is interesting since dropout is also known to prevent catastrophic forgetting [112] and thus in general it seems to help DNNs retain patterns.

A number of arguments support the idea that SGD-based learning imparts a regularization effect, especially with a small batch size [310] or a small number of epochs [131]. Previous work also suggests that SGD prioritizes the learning of simple hypothesis first. Sjoberg et al. [266] showed that, for linear models, SGD first learns models with small ℓ^2 parameter norm. More generally, the efficacy of early stopping shows that SGD first

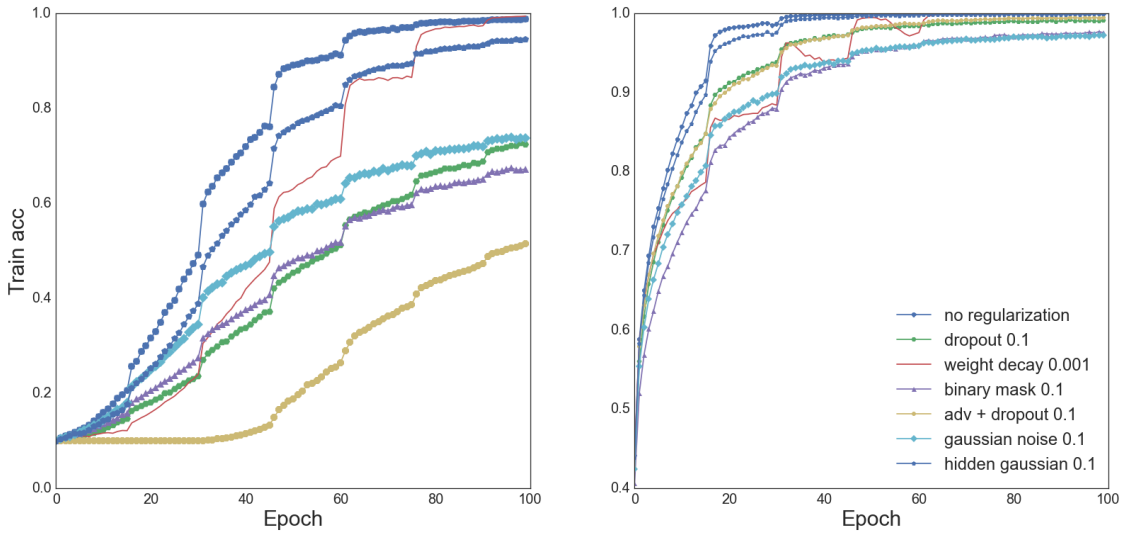


Figure 3.11: Training curves for different regularization techniques on random label (left) and real (right) data. The vertical ordering of the curves is different for random labels than for real data, indicating differences in the propensity of different regularizers to slow-down memorization.

learns simpler models [314]. We extend these results, showing that DNNs trained with SGD learn patterns before memorizing, *even in the presence of noise examples*.

Various previous works have analyzed explanations for the generalization power of DNNs. Montavon et al. [215] use kernel methods to analyze the complexity of deep learning architectures, and find that network priors (e.g. implemented by the network structure of a CNN or MLP) control the speed of learning at each layer. Neyshabur et al. [220] note that the number of parameters does not control the effective capacity of a DNN, and that the reason for DNNs' generalization is unknown. We supplement this result by showing how the impact of representational capacity changes with varying noise levels. While exploring the effect of noise samples on learning dynamics has a long tradition [6, 27], we are the first to examine *relationships* between the fraction of noise samples and other attributes of the learning algorithm, namely: capacity, training time and dataset size.

Multiple techniques for analyzing the training of DNNs have been proposed before, including looking at generalization error, trajectory length evolution [239], analyzing Jacobians associated to different layers [255, 302], or the shape of the loss minima

found by SGD [47, 155, 159]. Instead of measuring the sharpness of the loss for the learned hypothesis, we investigate the complexity of the learned hypothesis throughout training and across different datasets and regularizers, as measured by the critical sample ratio. Critical samples refer to real data-points that have adversarial examples [115, 275] nearby. Adversarial examples originally referred to imperceptibly perturbed data-points that are confidently misclassified. [213] define *virtual* adversarial examples via changes in the predictive distribution instead, thus extending the definition to unlabeled data-points. Kurakin et al. [180] recommend using this definition when training on adversarial examples, and it is the definition we use.

Two contemporary works perform in-depth explorations of topics related to our work. Bojanowski and Joulin [29] show that predicting random noise targets can yield state of the art results in unsupervised learning, corroborating our findings in subsection 3.3.1, especially Figure 3.3.1. Koh and Liang [170] use *influence functions* to measure the impact on parameter changes during training, as in our subsection 3.3.2. They explore several promising applications for this technique, including generation of adversarial *training* examples.

3.7 Conclusion

Our empirical exploration demonstrates qualitative differences in DNN optimization on noise vs. real data, all of which support the claim that DNNs trained with SGD-variants first use patterns, not brute force memorization, to fit real data. However, since DNNs have the demonstrated ability to fit noise, it is unclear why they find generalizable solutions on real data; we believe that the deep learning priors including distributed and hierarchical representations likely play an important role. Our analysis suggests that memorization and generalization in DNNs depend on network architecture and optimization procedure, but also on the data itself. We hope to encourage future research on how properties of datasets influence the behavior of deep learning algorithms, and suggest a data-dependent understanding of DNN capacity as a research goal.

Acknowledgments We thank Akram Erraqabi, Jason Jo and Ian Goodfellow for helpful discussions. SJ was supported by Grant No. DI 2014/016644 from Ministry of Science and Higher Education, Poland. DA was supported by IVADO, CIFAR and NSERC. EB was financially supported by the Samsung Advanced Institute of Technology (SAIT). MSK and SJ were supported by MILA during the course of this work. We acknowledge the computing resources provided by ComputeCanada and CalculQuebec. Experiments were carried out using Theano [282] and Keras [53].

CHAPTER 4

DISCUSSION OF "OUT-OF-DISTRIBUTION GENERALIZATION VIA RISK EXTRAPOLATION (REX)"

Author List

David Krueger
Mila;
University of Montreal

Jonathan Binas
Mila;
University of Montreal

Ethan Caballero
Mila;
University of Montreal

Dinghuai Zhang
Mila;
University of Montreal

Joern-Henrik Jacobsen
Vector;
University of Toronto

Remi Le Priol
Mila;
University of Montreal

Amy Zhang
Mila;
McGill University;
Facebook AI Research

Aaron Courville
Mila;
University of Montreal

4.1 My Contributions

1. I conceived of the method(s) and the project.
2. I managed the project.
3. I wrote code for initial proof-of-concept experiments.
4. I did the large majority of the writing and conceived and created most of the illustrative figures.
5. I proved the theoretical results.
6. I helped conceive and plan the experiments.
7. I ran preliminary experiments on CMNIST and final experiments on toy SEMs.

4.2 Background

This background section describes the issue of robustness in deep learning, and summarizes relevant lines of work.

Despite generalizing very well on a variety of tasks and domains, neural networks – and indeed, machine learning more generally – are also quite brittle. Many surprising and poorly-understood failure modes have been demonstrated, and much research has been devoted to understanding and addressing such issues. While neural networks capabilities have certainly surprised many deep learning "skeptics", deep learning "believers" have been surprised by persistence and pervasiveness of such failure modes.

A recent approach to this problem, which is the topic of our work, is invariant prediction. Before discussing invariant prediction, however, we give a brief overview of other relevant research areas.

4.2.1 An overview of other areas of robustness research

Here, we give a brief overview of the following five lines of research relevant towards this topic:

1. domain generalization
2. Adversarial examples and adversarial robustness
3. Out-of-distribution (OOD) detection
4. systematic generalization
5. non-adversarial robustness

These are listed in the rough chronological order of their development as prominent areas of research.

Domain generalization refers to a problem setting where a model trained on several different **domains** (i.e. data distributions) is expected to generalize to a new domain(s) without any data from that domain. This field predates the publication of AlexNet [174],

and the resulting "deep learning revolution". Classic works in domain generalization often take a theoretical approach, examining what sort of assumptions are necessary or sufficient to achieve generalization to new domains. In particular, it is popular to assume that $P(Y|X)$ is fixed across domains, and that the distribution of $P(X)$ is not too different across domains.

Adversarial examples were the first clear sign that Deep Learning models were not robust. These were originally conceived as imperceptible perturbations to real data-points carefully chosen to fool a given model. While adversarial examples are a very specific kind of OOD data, the ability to generalize to adversarial examples may correlate with other forms of OOD generalization [92, 137].

Out-of-distribution (OOD) detection was popularized in Deep Learning by Hendrycks and Gimpel [139], and takes the more modest aim of *detecting* when a test example is from a different domain, rather than generalizing correctly to such novel examples (as in domain generalization). Despite being easier, this can still be a challenging problem, although modern approaches, specifically large transformer models, perform much better [94]. OOD detection is not an approach to OOD generalization, but rather an alternative. From the point-of-view of AI Alignment, it is an appealing one, because users are often content with a system that merely avoids catastrophic failures, e.g. by detecting when it might be liable to make a mistake and engaging safe default behavior.

Non-adversarial robustness of deep computer vision models, surprisingly, became a focus of Deep Learning research only after several years of intense interest in adversarial robustness. This line of work is closely related to OOD generalization, but more focused on examining and improving inductive biases of deep learning, generalizing from a single training domain, and OOD data that is more similar to the training data, e.g. images corrupted with noise, or collected under very similar conditions. Thus, robustness can be seen as a somewhat specific, and somewhat less ambitious, form of OOD generalization.

Systematic Generalization is the use of compositionality to generalize to new combinations of known concepts. This could enable out-of-distribution generalization to combinations which do not appear in the training domains. As noted in Section 1.4.1, Deep Learning has been motivated as an approach to systematic generalization, via disen-

tangling underlying factors of variation. However, as Bahdanau et al. [14] and Schott et al. [257] have shown, Deep Learning methods typically fail to generalize systematically. Thus it remains unclear how much successful systematic generalization would contribute to solving OOD generalization problems.

4.2.2 Invariant prediction

Given the existence of many predictive features, and many solutions with roughly 0 training loss, how are we to pick out solutions that robustly track human judgments or other sources of ground-truth? Invariant Risk Minimization (IRM) [8] provides one answer to this question, based on the notion of **invariant prediction**, as introduced by Peters et al. [233] in the context of causal discovery.

Invariant prediction is a method for learning predictive models, i.e. learning about the *directional* relationship between between X and Y in terms of a predictive distribution $\hat{P}(Y|X)$. The intuition underlying invariant prediction is this:

- **A robust model is one that works well in many different circumstances.** These different circumstances are classically referred to as **domains**, although Arjovsky et al. [8] use **environments** to refer to the same concept. Mathematically, the difference between domains comes down to a difference in the joint distribution of the data $P(X,Y)$.

- **Robustness thus requires learning about relationships between X and Y that are consistent across different circumstances.** For instance, causal relationships are consistent in this sense: if X is the direct cause of Y, then $P(Y|X)$ will the same regardless of external circumstances – so long as there is no direct intervention on the value of Y, or selection process affecting the data.

- **By biasing models to learn relationships that appear consistent, we can improve model robustness.** Although we may not have access all of the data we would need in order to assess whether a given relationship is consistent, we can still compare different models based on how consistent they are across different domains that we do observe.

While invariant prediction may seems like a natural desiderata, in the roughly 2 years since Arjovsky et al. [8], there has so far been limited success in applying this principle to solve the problems of robustness observed in deep network. The only notable success I'm currently aware of is Wald et al. [300], who encourage models to be calibrated on each domain, after noting that this is a condition for achieving invariant prediction. Overall, it seems likely that other novel inductive biases may play an equally significant or even more significant role than invariant prediction.

In looking at variation across different domains as a guide for how to generalize robustly to a novel test domain, IRM and REx are both forms of domain generalization. REx develops an alternative to IRM, which can perform invariant prediction, but which also behaves qualitatively differently in some settings. This difference in behavior can be (dis)advantageous, depending on the particular task at hand.

4.3 Contributions of the work

This work provided a new perspective on invariant prediction as a robust optimization problem; this is especially significant since Arjovsky et al. [8] contrast Invariant Risk Minimization with robust optimization approaches. This helped to popularize the study of invariant prediction in deep learning. Our theoretical contributions are best summarized by the Remark, which explains why optimizing performance on a large, diverse dataset should not be expected to solve OOD generalization problems. We also introduced novel arguments for the limitations of invariant prediction to handle different kinds of distributions shift. Finally, we called attention to methodological issues in evaluating OOD generalization; a subsequent in-depth study found that standard ERM training outperformed customized OOD generalization techniques when using proper methodology [123].

CHAPTER 5

OUT-OF-DISTRIBUTION GENERALIZATION VIA RISK EXTRAPOLATION (REX)

ABSTRACT

Distributional shift is one of the major obstacles when transferring machine learning prediction systems from the lab to the real world. To tackle this problem, we assume that variation across training domains is representative of the variation we might encounter at test time, but also that *shifts at test time may be more extreme in magnitude*. In particular, we show that reducing differences in risk across training domains can reduce a model's sensitivity to a wide range of extreme distributional shifts, including the challenging setting where the input contains both causal and anti-causal elements. We motivate this approach, **Risk Extrapolation (REx)**, as a form of robust optimization over a perturbation set of extrapolated domains (MM-REx), and propose a penalty on the variance of training risks (V-REx) as a simpler variant. We prove that variants of REx can recover the causal mechanisms of the targets, while also providing some robustness to changes in the input distribution ("covariate shift"). By trading-off robustness to causally induced distributional shifts and covariate shift, REx is able to outperform alternative methods such as Invariant Risk Minimization in situations where these types of shift co-occur.

5.1 Introduction

While neural networks often exhibit super-human generalization on the training distribution, they can be extremely sensitive to distributional shift, presenting a major roadblock for their practical application [87, 137, 241, 273]. This sensitivity is often caused by relying on "spurious" features unrelated to the core concept we are trying to learn [103]. For instance, Beery et al. [16] give the example of an image recognition model failing to correctly classify cows on the beach, since it has learned to make

predictions based on the features of the background (e.g. a grassy field) instead of just the animal.

In this work, we consider **out-of-distribution (OOD) generalization**, also known as **domain generalization**, where a model must generalize appropriately to a new test domain for which it has neither labeled nor unlabeled training data. Following common practice [19], we formulate this as optimizing the worst-case performance over a **perturbation set** of possible test domains, $\mathscr{F}$:

$$\mathbb{R}^{\text{OOD}}_{\mathscr{F}}(\theta) = \max_{e \in \mathscr{F}} \mathbb{R}_e(\theta) \tag{5.1}$$

Since generalizing to arbitrary test domains is impossible, the choice of perturbation set encodes our assumptions about which test domains might be encountered. Instead of making such assumptions *a priori*, we assume access to data from multiple training domains, which can inform our choice of perturbation set. A classic approach for this setting is **group distributionally robust optimization (DRO)** [250], where $\mathscr{F}$ contains all mixtures of the training distributions. This is mathematically equivalent to considering convex combinations of the training *risks*.

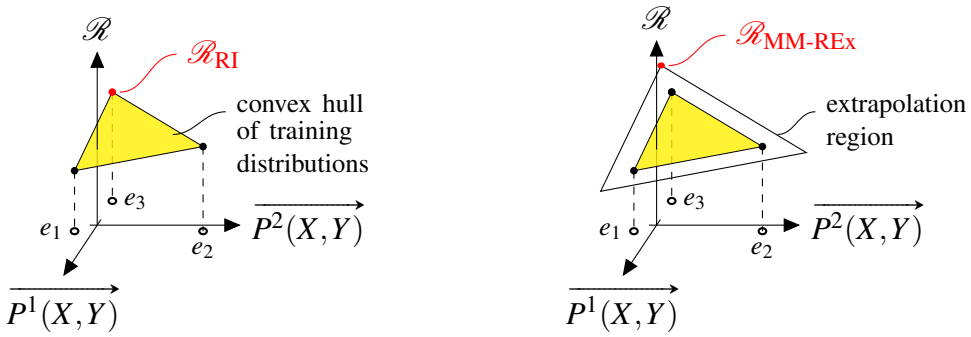


Figure 5.1: **Left**: Robust optimization optimizes worst-case performance over the convex hull of training distributions. **Right**: By extrapolating risks, REx encourages robustness to larger shifts. Here e_1, e_2, and e_3 represent training distributions, and $\overrightarrow{P^1(X,Y)}$, $\overrightarrow{P^2(X,Y)}$ represent some particular directions of variation in the affine space of quasiprobability distributions over (X,Y).

However, we aim for a more ambitious form of OOD generalization, over a larger

perturbation set. Our method **minimax Risk Extrapolation (MM-REx)** is an extension of DRO where $\mathscr{F}$ instead contains *affine* combinations of training risks, see Figure 5.1. Under specific circumstances, MM-REx can be thought of as DRO over a set of extrapolated domains.[1] But MM-REx also unlocks fundamental new generalization capabilities unavailable to DRO.

In particular, focusing on supervised learning, we show that Risk Extrapolation can uncover invariant relationships between inputs X and targets Y. Intuitively, an **invariant relationship** is a statistical relationship which is maintained across all domains in $\mathscr{F}$. Returning to the cow-on-the-beach example, the relationship between the animal and the label is expected to be invariant, while the relationship between the background and the label is not. A model which bases its predictions on such an invariant relationship is said to perform **invariant prediction**.[2]

Many domain generalization methods assume $P(Y|X)$ is an invariant relationship, limiting distributional shift to changes in $P(X)$, which are known as **covariate shift** [18]. This assumption can easily be violated, however. For instance, when Y causes X, a more sensible assumption is that $P(X|Y)$ is fixed, with $P(Y)$ varying across domains [198, 256]. In general, invariant prediction may involve an aspect of causal discovery. Depending on the perturbation set, however, other, more predictive, invariant relationships may also exist [171].

The first method for invariant prediction to be compatible with modern deep learning problems and techniques is **Invariant Risk Minimization (IRM)** [8], making it a natural point of comparison. Our work focuses on explaining how REx addresses OOD generalization, and highlighting differences (especially advantages) of REx compared with IRM and other domain generalization methods, see Table 5.I. Broadly speaking, REx optimizes for robustness to the forms of distributional shift that have been observed to have the largest impact on performance in training domains. This can be a significant advantage over the more focused (but also limited) robustness that IRM targets. For instance, unlike IRM, REx can also encourage robustness to covariate shift (see Section 5.3

[1] We define "extrapolation" to mean "outside the convex hull", see Appendix 11.1.2 for more.

[2] Note this is different from learning an invariant representation [100]; see Section 5.2.3.

and Figure 5.3.4).

Our experiments show that REx significantly outperforms IRM in settings that involve covariate shift and require invariant prediction, including modified versions of CMNIST and simulated robotics tasks from the Deepmind control suite. On the other hand, because REx does not distinguish between underfitting and inherent noise, IRM has an advantage in settings where some domains are intrinsically harder than others. Our contributions include:

1. MM-REx, a novel domain generalization problem formulation suitable for invariant prediction.

2. Demonstrating that REx solves invariant prediction tasks where IRM fails due to covariate shift.

3. Proving that equality of risks can be a sufficient criteria for discovering causal structure.

Method	Invariant Prediction	Cov. Shift Robustness	Suitable for Deep Learning
DRO	✗	✓	✓
(C-)ADA	✗	✓	✓
ICP	✓	✗	✗
IRM	✓	✗	✓
REx	✓	✓	✓

Table 5.I: A comparison of approaches for OOD generalization.

5.2 Background & related work

We consider multi-source domain generalization, where our goal is to find parameters θ that perform well on unseen domains, given a set of m training **domains**, $\mathscr{E} = \{e_1, .., e_m\}$, sometimes also called **environments**. We assume the loss function,

ℓ is fixed, and domains only differ in terms of their data distribution $P_e(X,Y)$ and dataset D_e. The **risk function** for a given domain/distribution e is:

$$\mathbb{R}_e(\theta) \doteq \mathbb{E}_{(x,y)\sim P_e(X,Y)}\, \ell(f_\theta(x),y) \tag{5.2}$$

We refer to members of the set $\{\mathbb{R}_e \,|e \in \mathscr{E}\}$ as the **training risks** or simply **risks**. Changes in $P_e(X,Y)$ can be categorized as either changes in $P(X)$ (**covariate shift**), changes in $P(Y|X)$ (**concept shift**), or a combination. The standard approach to learning problems is **Empirical Risk Minimization (ERM)**, which minimizes the average loss across all the training examples from all the domains:

$$\mathbb{R}_{\text{ERM}}(\theta) \doteq \mathbb{E}_{(x,y)\sim \cup_{e\in\mathscr{E}} D_e}\, \ell(f_\theta(x),y) \tag{5.3}$$

$$= \sum_e |D_e|\, \mathbb{E}_{(x,y)\sim D_e}\, \ell(f_\theta(x),y) \tag{5.4}$$

5.2.1 Robust optimization

An approach more tailored to OOD generalization is **robust optimization** [19], which aims to optimize a model's worst-case performance over some **perturbation set** of possible data distributions, $\mathscr{F}$ (see Eqn. 5.1). When only a single training domain is available (**single-source domain generalization**), it is common to assume that $P(Y|X)$ is fixed, and let $\mathscr{F}$ be all distributions within some f-divergence ball of the training $P(X)$ [12, 147]. As another example, adversarial robustness can be seen as instead using a Wasserstein ball as a perturbation set [264]. The assumption that $P(Y|X)$ is fixed is commonly called the "covariate shift assumption" [18]; however, we assume that covariate shift and concept shift can co-occur, and refer to this assumption as **the fixed relationship assumption (FRA)**.

In **multi-source domain generalization**, test distributions are often assumed to be mixtures (i.e. convex combinations) of the training distributions; this is equivalent to

setting $\mathscr{F} \doteq \mathscr{E}$:

$$\mathbb{R}_{\text{RI}}(\theta) \doteq \max_{\substack{\Sigma_e \lambda_e = 1 \\ \lambda_e \geq 0}} \sum_{e=1}^{m} \lambda_e \mathbb{R}_e(\theta) = \max_{e \in \mathscr{E}} \mathbb{R}_e(\theta). \tag{5.5}$$

We call this objective **Risk Interpolation (RI)**, or, following Sagawa et al. [250], **(group) Distributionally Robust Optimization (DRO)**. While single-source methods classically assume that the probability of each data-point can vary independently [147], DRO yields a much lower dimensional perturbation set, with at most one direction of variation per domain, regardless of the dimensionality of X and Y. It also does not rely on FRA, and can provide robustness to any form of shift in $P(X,Y)$ which occurs across training domains. Minimax-REx is an extension of this approach to affine combinations of training risks.

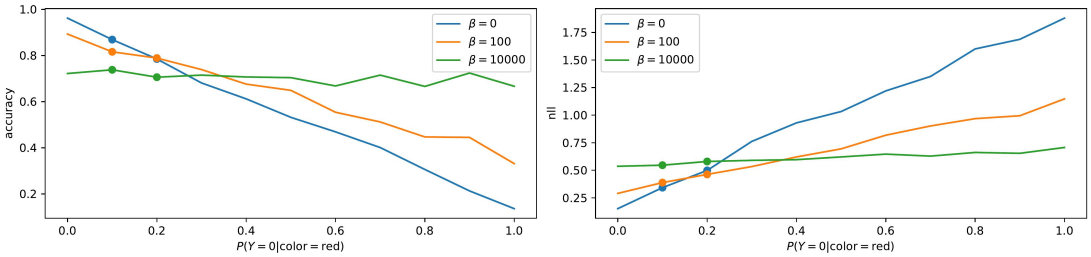


Figure 5.2: Training accuracies (**left**) and risks (**right**) on colored MNIST domains with varying $P(Y=0|\text{color}=\text{red})$ after 500 epochs. Dots represent training risks, lines represent test risks on different domains. Increasing the V-REx penalty (β) leads to a flatter "risk plane" and more consistent performance across domains, as the model learns to ignore color in favor of shape-based invariant prediction. Note that $\beta = 100$ gives the best worst-case risk across the 2 training domains, and so would be the solution preferred by DRO [250]. This demonstrates that REx's counter-intuitive propensity to *increase* training risks can be necessary for good OOD performance.

5.2.2 Invariant representations vs. invariant predictors

An **equipredictive representation**, Φ, is a function of X with the property that $P_e(Y|\Phi)$ is equal, $\forall e \in \mathscr{F}$. In other words, the relationship between such a Φ and Y is fixed across domains. **Invariant relationships** between X and Y are then exactly those that can be written as $P(Y|\Phi(x))$ with Φ an equipredictive representation. A model

$\hat{P}(Y|X=x)$ that learns such an invariant relationship is called an **invariant predictor**. Intuitively, an invariant predictor works equally well across all domains in $\mathscr{F}$. The principle of risk extrapolation aims to achieve invariant prediction by enforcing such equality across *training* domains $\mathscr{E}$, and does not rely on explicitly learning an equipredictive representation.

Koyama and Yamaguchi [171] prove that a *maximal* equipredictive representation – that is, one that maximizes mutual information with the targets, $\Phi^* \doteq \text{argmax}_\Phi I(\Phi, Y)$ – solves the robust optimization problem (Eqn. 5.1) under fairly general assumptions.[3] When Φ^* is unique, we call the features it ignores **spurious**. The result of Koyama and Yamaguchi [171] provides a theoretical reason for favoring invariant prediction over the common approach of learning **invariant *representations*** [229], which make $P_e(\Phi)$ or $P_e(\Phi|Y)$ equal $\forall e \in \mathscr{E}$. Popular methods here include **adversarial domain adaptation (ADA)** [100] and **conditional ADA (C-ADA)** [201]. Unlike invariant predictors, invariant representations can easily fail to generalize OOD: ADA forces the predictor to have the same marginal predictions $\hat{P}(Y)$, which is a mistake when $P(Y)$ in fact changes across domains [319]; C-ADA suffers from more subtle issues [8].

5.2.3 Invariance and causality

The relationship between cause and effect is a paradigmatic example of an invariant relationship. Here, we summarize definitions from causal modeling, and discuss causal approaches to domain generalization. We will refer to these definitions for the statements of our theorems in Section 5.3.4.

5.2.3.0.1 Definitions. A **causal graph** is a directed acyclic graph (DAG), where nodes represent variables and edges point from causes to effects. In this work, we use **Structural Causal Models (SCMs)**, which also specify how the value of a variable is computed given its parents. An SCM, $\mathfrak{C}$, is defined by specifying the **mechanism**, $f_Z : Pa(Z) \to dom(Z)$

[3] The first formal definition of an equipredictive representation we found was by Koyama and Yamaguchi [171], who use the term "(maximal) invariant predictor". We prefer our terminology since: 1) it is more consistent with Arjovsky et al. [8], and 2) Φ is a representation, not a predictor.

for each variable Z.[4] Mechanisms are *deterministic*; noise in Z is represented explicitly via a special noise variable N_Z, and these noise variables are jointly independent. An **intervention**, ι is any modification to the mechanisms of one or more variables; an intervention can introduce new edges, so long as it does not introduce a cycle. $do(X_i = x)$ denotes an intervention which sets X_i to the constant value x (removing all incoming edges). Data can be generated from an SCM, $\mathfrak{C}$, by sampling all of the noise variables, and then using the mechanisms to compute the value of every node whose parents' values are known. This sampling process defines an **entailed distribution**, $P^{\mathfrak{C}}(\mathbf{Z})$ over the nodes $\mathbf{Z}$ of $\mathfrak{C}$. We overload f_Z, letting $f_Z(\mathbf{Z})$ refer to the conditional distribution $P^{\mathfrak{C}}(Z|\mathbf{Z} \setminus \{Z\})$.

5.2.4 Causal approaches to domain generalization

Instead of assuming $P(Y|X)$ is fixed (FRA), works that take a causal approach to domain generalization often assume that the *mechanism* for Y is fixed; we call this **the fixed mechanism assumption (FMA)**. Meanwhile, they assume X may be subject to different (e.g. arbitrary) interventions in different domains [43]. We call changes in $P(X,Y)$ resulting from interventions on X **interventional shift**. Interventional shift can involve both covariate shift and/or concept shift. In their seminal work on **Invariant Causal Prediction (ICP)**, Peters et al. [233] leverage this invariance to learn which elements of X cause Y. ICP and its nonlinear extension [136] use statistical tests to detect whether the residuals of a linear model are equal across domains. Our work differs from ICP in that:

1. Our method is model agnostic and scales to deep networks.
2. Our goal is OOD generalization, not causal inference. These are not identical: invariant prediction can sometimes make use of non-causal relationships, but when deciding which interventions to perform, a truly causal model is called for.
3. Our learning principle only requires invariance of risks, not residuals. Nonetheless, we prove that this can ensure invariant causal prediction.

[4] Our definitions follow *Elements of Causal Inference* [234]; our notation mostly does as well.

A more similar method to REx is **Invariant Risk Minimization (IRM)** [8], which shares properties (1) and (2) of the list above. Like REx, IRM also uses a weaker form of invariance than ICP; namely, they insist that the optimal linear classifier must match across domains.[5] Still, REx differs significantly from IRM. While IRM specifically aims for invariant prediction, REx seeks robustness to *whichever* forms of distributional shift are present. Thus, REx is more directly focused on the problem of OOD generalization, and can provide robustness to a wider variety of distributional shifts, inluding covariate shift. Also, unlike REx, IRM seeks to match $\mathbb{E}(Y|\Phi(X))$ across domains, not the full $P(Y|\Phi(X))$. This, combined with IRM's indifference to covariate shift, make it more effective in cases where different domains or examples are inherently more noisy.

5.2.5 Fairness

Equalizing risk across different groups (e.g. male vs. female) has been proposed as a definition of **fairness** [79], generalizing the equal opportunity definition of fairness [132]. Williamson and Menon [309] propose using the absolute difference of risks to measure deviation from this notion of fairness; this corresponds to our MM-REx, in the case of only two domains, and is similar to V-REx, which uses the variance of risks. However, in the context of fairness, equalizing the risk of training groups is the *goal*. Our work goes beyond this by showing that it can serve as a *method* for OOD generalization.

5.3 Risk extrapolation

Before discussing algorithms for REx and theoretical results, we first expand on our high-level explanations of what REx does, what kind of OOD generalization it promotes, and how. The principle of Risk Extrapolation (REx) has two aims:

1. Reducing training risks

2. Increasing similarity of training risks

[5] In practice, IRMv1 replaces this bilevel optimization problem with a gradient penalty on classifier weights.

In general, these goals can be at odds with each other; decreasing the risk in the domain with the lowest risk also decreases the overall similarity of training risks. Thus methods for REx may seek to *increase* risk on the best performing domains. While this is counter-intuitive, it can be necessary to achieve good OOD generalization, as Figure 5.2 demonstrates. From a geometric point of view, encouraging equality of risks flattens the "risk plane" (the affine span of the training risks, considered as a function of the data distribution, see Figures 5.1 and 5.2). While this can result in higher training risks, it also means that the risk changes less if the distributional shifts between training domains are magnified at test time.

Figure 5.2 illustrates how flattening the risk plane can promote OOD generalization on real data, using the Colored MNIST (CMNIST) task as an example [8]. In the CMNIST training domains, the color of a digit is more predictive of the label than the shape is. But because the correlation between color and label is not invariant, predictors that use the color feature achieve different risk on different domains. By enforcing equality of risks, REx prevents the model from using the color feature enabling successful generalization to the test domain where the correlation between color and label is reversed.

5.3.1 Probabilities vs. risks

Figure 5.3 depicts how the extrapolated risks considered in MM-REx can be translated into a corresponding change in $P(X,Y)$, using an example of pure covariate shift. Training distributions can be thought of as points in an affine space with a dimension for every possible value of (X,Y); see Appendix 11.1.4 for an example. Because the risk is linear w.r.t. $P(\mathbf{x},\mathbf{y})$, a convex combination of risks from different domains is equivalent to the risk on a domain given by the mixture of their distributions. The same holds for the affine combinations used in MM-REx, with the caveat that the negative coefficients may lead to negative probabilities, making the resulting $P(X,Y)$ a ***quasi*probability distribution**, i.e. a signed measure with integral 1. We explore the theoretical implications of this in Appendix 11.1.7.

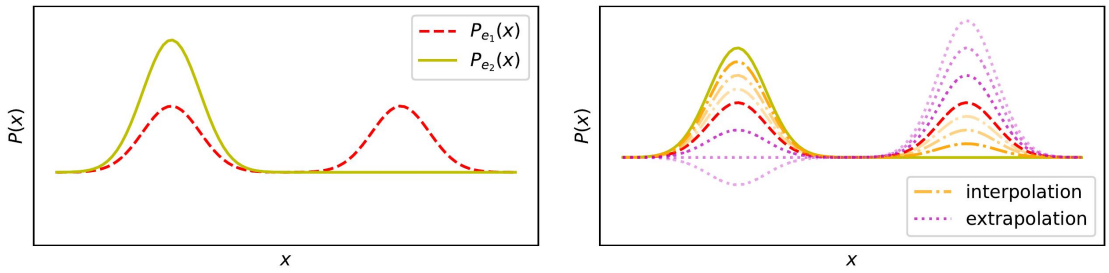


Figure 5.3: Extrapolation can yield a distribution with *negative* $P(x)$ for some x. **Left:** $P(x)$ for domains e_1 and e_2. **Right:** Point-wise interpolation/extrapolation of $P^{e_1}(x)$ and $P^{e_2}(x)$. Since MM-REx target worst-case robustness across extrapolated domains, it can provide robustness to such shifts in P(X) (covariate shift).

5.3.2 Covariate shift

When only $P(X)$ differs across domains (i.e. FRA holds), as in Figure 5.3, then $\Phi(x) = x$ is already an equipredictive representation, and so *any* predictor is an invariant predictor. Thus methods which only promote invariant prediction – such as IRM – are not expected to improve OOD generalization (compared with ERM). Indeed, Arjovsky et al. [8] recognize this limitation of IRM in what they call the "realizable" case. Instead, what is needed is robustness to covariate shift, which REx, but not IRM, can provide. Robustness to covariate shift can improve OOD generalization by ensuring that low-capacity models spend sufficient capacity on low-density regions of the input space; we show how REx can provide such benefits in Appendix 11.1.5. But even for high capacity models, $P(X)$ can have a significant influence on what is learned; for instance Sagawa et al. [250] show that DRO can significantly improves the performance on rare groups for a model that achieves 100% training accuracy in their Waterbirds dataset. Pursuing robustness to covariate shift also comes with drawbacks for REx, however: REx does not distinguish between underfitting and inherent noise in the data, and so can force the model to make equally bad predictions everywhere, even if some examples are less noisy than others.

5.3.3 Methods of risk extrapolation

We now formally describe the **Minimax REx (MM-REx)** and **Variance-REx (V-REx)** techniques for risk extrapolation. Minimax-REx performs robust learning over a perturbation set of *affine* combinations of training risks with bounded coefficients:

$$\mathbb{R}_{\text{MM-REx}}(\theta) \doteq \max_{\substack{\Sigma_e \lambda_e = 1 \\ \lambda_e \geq \lambda_{\min}}} \sum_{e=1}^{m} \lambda_e \mathbb{R}_e(\theta) \tag{5.6}$$

$$= (1 - m\lambda_{\min}) \max_e \mathbb{R}_e(\theta) + \lambda_{\min} \sum_{e=1}^{m} \mathbb{R}_e(\theta), \tag{5.7}$$

where m is the number of domains, and the hyperparameter $\lambda_{\min}$ controls how much we extrapolate. For negative values of $\lambda_{\min}$, MM-REx places negative weights on the risk of all but the worst-case domain, and as $\lambda_{\min} \to -\infty$, this criterion enforces strict equality between training risks; $\lambda_{\min} = 0$ recovers risk interpolation (RI). Thus, like RI, MM-REx aims to be robust in the direction of variations in $P(X,Y)$ between test domains. However, negative coefficients allow us to extrapolate to more extreme variations. Geometrically, larger values of $\lambda_{\min}$ expand the perturbation set farther away from the convex hull of the training risks, encouraging a flatter "risk-plane" (see Figure 5.2).

While MM-REx makes the relationship to RI/RO clear, we found using the variance of risks as a regularizer (V-REx) simpler, stabler, and more effective:

$$\mathbb{R}_{\text{V-REx}}(\theta) \doteq \beta \ \text{Var}(\{\mathbb{R}_1(\theta), ..., \mathbb{R}_m(\theta)\}) + \sum_{e=1}^{m} \mathbb{R}_e(\theta) \tag{5.8}$$

Here $\beta \in [0, \infty)$ controls the balance between reducing average risk and enforcing equality of risks, with $\beta = 0$ recovering ERM, and $\beta \to \infty$ leading V-REx to focus entirely on making the risks equal. See Appendix for the relationship between V-REx and MM-REx and their gradient vector fields.

5.3.4 Theoretical conditions for REx to perform causal discovery

We now prove that exactly equalizing training risks (as incentivized by REx) leads a model to learn the causal mechanism of Y under assumptions similar to those of Peters et al. [233], namely:

1. The causes of Y are observed, i.e. $Pa(Y) \subseteq X$.

2. Domains correspond to interventions on X.

3. Homoskedasticity (a slight generalization of the additive noise setting assumed by Peters et al. [233]). We say an SEM $\mathfrak{C}$ is **homoskedastic** (with respect to a loss function ℓ), if the Bayes error rate of $\ell(f_Y(x), f_Y(x))$ is the same for all $x \in \mathscr{X}$.[6]

The contribution of our theory (vs. ICP) is to prove that equalizing risks is sufficient to learn the causes of Y. In contrast, they insist that the entire distribution of error residuals (in predicting Y) be the same across domains. We provide proof sketches here and complete proofs in the appendix.

Theorem 1 demonstrates a practical result: we can identify a linear SEM model using REx with a number of domains linear in the dimensionality of X.

Theorem 1. *Given a Linear SEM, $X_i \leftarrow \sum_{j \neq i} \beta_{(i,j)} X_j + \varepsilon_i$, with $Y \doteq X_0$, and a predictor $f_\beta(X) \doteq \sum_{j:j>0} \beta_j X_j + \varepsilon_j$ that satisfies REx (with mean-squared error) over a perturbation set of domains that contains 3 distinct $do()$ interventions for each $X_i : i > 0$. Then $\beta_j = \beta_{0,j}, \forall j$.*

Proof Sketch. We adapt the proof of Theorem 4i from Peters et al. [233]. They show that matching the residual errors across observational and interventional domains forces the model to learn f_Y. We use the weaker condition of matching risks to derive a quadratic equation that the $do()$ interventions must satisfy for any model other than f_Y. Since there

[6] Note that our definitions of **homoskedastic/heteroskedastic** do *not* correspond to the types of domains constructed in Arjovsky et al. [8], Section 5.1, but rather are a generalization of the definitions of these terms as commonly used in statistics. Specifically, for us, *hetero*skedasticity means that the "predicatability" (e.g. variance) of Y differs across inputs x, whereas for Arjovsky et al. [8], it means the predicatability of Y at a given input varies across *domains*; we refer to this second type as *domain*-homo/heteroskedasticity for clarity.

are at most 2 solutions to a quadratic equation, insisting on equality of risks across 3 distinct $do()$ interventions forces the model to learn f_Y.

Given the assumption that a predictor satisfies REx over *all* interventions that do not change the mechanism of Y, we can prove a much more general result. We now consider an arbitrary SCM, $\mathfrak{C}$, generating Y and X, and let $\mathscr{E}^I$ be the set of domains corresponding to arbitrary interventions on X, similarly to Peters et al. [233].

Theorem 2. *Suppose ℓ is a (strictly) proper scoring rule. Then a predictor that satisfies REx for a over $\mathscr{E}^I$ uses $f_Y(x)$ as its predictive distribution on input x for all $x \in \mathscr{X}$.*

Proof Sketch. Since the distribution of Y given its parents doesn't depend on the domain, f_Y can make reliable point-wise predictions across domains. This translates into equality of risk across domains when the overall difficulty of the examples is held constant across domains, e.g. by assuming homoskedasticity.[7] While a different predictor might do a better job on *some* domains, we can always find an domain where it does worse than f_Y, and so f_Y is both unique and optimal.

Remark. Theorem 2 is only meant to provide insight into how the REx principle relates to causal invariance; the perturbation set in this theorem is uncountably infinite. Note, however, that even in this setting, the ERM principle does *not*, in general, recover the causal mechanism for Y. Rather, the ERM solution depends on the distribution over domains. For instance, if all but an $\varepsilon \to 0$ fraction of the data comes from the CMNIST training domains, then ERM will learn to use the color feature, just as in original the CMNIST task.

5.4 Experiments

We evaluate REx and compare with IRM on a range of tasks requiring OOD generalization. REx provides generalization benefits and outperforms IRM on a wide range of tasks, including: i) variants of the Colored MNIST (CMNIST) dataset [8] with covariate shift, ii) continuous control tasks with partial observability and spurious features, iii)

[7] Note we could also assume no covariate shift in order to fix the difficulty, but this seems hard to motivate in the context of interventions on X, which can change $P(X)$.

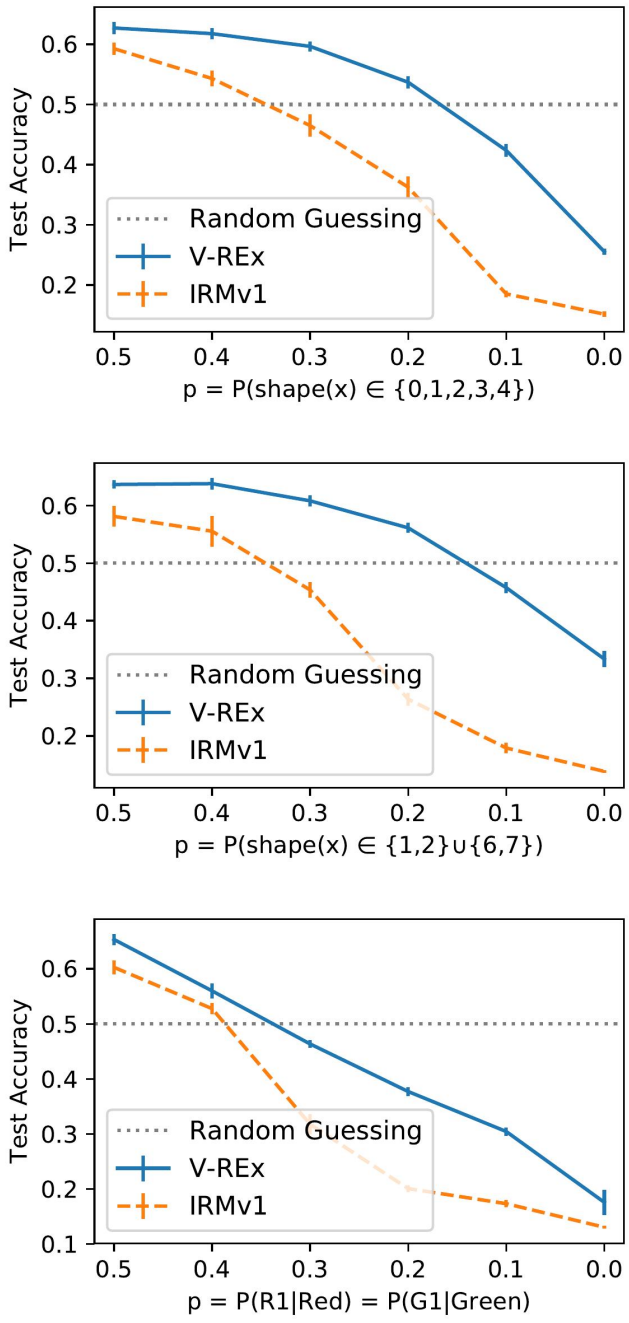


Figure 5.4: REx outperforms IRM on Colored MNIST variants that include covariate shift. The x-axis indexes increasing amount of shift between training distributions, with $p = 0$ corresponding to disjoint supports. **Left:** class imbalance, **Center:** shape imbalance, **Right:** color imbalance.

domain generalization tasks from the DomainBed suite [123]. On the other hand, when the inherent noise in Y varies across environments, IRM succeeds and REx performs poorly.

5.4.1 Colored MNIST

Arjovsky et al. [8] construct a binary classification problem (with 0-4 and 5-9 each collapsed into a single class) based on the MNIST dataset, using color as a spurious feature. Specifically, digits are either colored red or green, and there is a strong correlation between color and label, which is reversed at test time. The goal is to learn the causal "digit shape" feature and ignore the anti-causal "digit color" feature. The learner has access to three domains:

1. A training domain where green digits have a 80% chance of belonging to class 1 (digits 5-9).
2. A training domain where green digits have a 90% chance of belonging to class 1.
3. A test domain where green digits have a 10% chance of belonging to class 1.

Method	**train acc**	**test acc**
V-REx (ours)	71.5 ± 1.0	~~**68.7 ± 0.9**~~
IRM	70.8 ± 0.9	~~66.9 ± 2.5~~
MM-REx (ours)	72.4 ± 1.8	~~66.1 ± 1.5~~
RI	88.9 ± 0.3	~~22.3 ± 4.6~~
ERM	87.4 ± 0.2	~~17.1 ± 0.6~~
Grayscale oracle	73.5 ± 0.2	~~73.0 ± 0.4~~
Optimum	75	75
Chance	50	50

Table 5.II: Accuracy (percent) on Colored MNIST. REx and IRM learn to ignore the spurious color feature. ~~Strikethrough~~ results achieved via tuning on the test set.

We use the exact same hyperparameters as Arjovsky et al. [8], only replacing the

IRMv1 penalty with MM-REx or V-REx penalty.[8] These methods all achieve similar performance, see Table 5.II.

CMNIST with covariate shift. To test our hypothesis that REx should outperform IRM under covariate shift, we construct 3 variants of the CMNIST dataset. Each variant represents a different way of inducing covariate shift to ensure differences across methods are consistent. These experiments combine covariate shift with interventional shift, since $P(Green|Y=1)$ still differs across training domains as in the original CMNIST.

1. **Class imbalance:** varying $p = P(\text{shape(x)} \in \{0,1,2,3,4\})$; as in Wu et al. [311].
2. **Digit imbalance:** varying $p = P(\text{shape(x)} \in \{1,2\} \cup \{6,7\})$; digits 0 and 5 are removed.
3. **Color imbalance:** We use 2 versions of each color, for 4 total channels: R_1, R_2, G_1, G_2. We vary $p = P(R_1|Red) = P(G_1|Green)$.

While (1) also induces change in $P(Y)$, (2) and (3) induce *only* covariate shift in the causal shape and anti-causal color features (respectively). We compare across several levels of imbalance, $p \in [0, 0.5]$, using the same hyperparameters from Arjovsky et al. [8], and plot the mean and standard error over 3 trials.

V-REx significantly outperforms IRM in every case, see Figure 5.3.4. In order to verify that these results are not due to bad hyperparameters for IRM, we perform a random search that samples 340 unique hyperparameter combinations for each value of p, and compare the the number of times each method achieves better than chance-level (50% accuracy). Again, V-REx outperforms IRM; in particular, for small values of p, IRM never achieves better than random chance performance, while REx does better than random in 4.4%/23.7%/2.0% of trials, respectively, in the class/digit/color imbalance scenarios for $p = 0.1/0.1/0.2$. This indicates that REx can achieve good OOD generalization in settings involving both covariate and interventional shift, whereas IRM struggles to do so.

[8] When there are only 2 domains, MM-REx is equivalent to a penalty on the Mean Absolute Error (MAE), see Appendix 11.1.8.2.2.

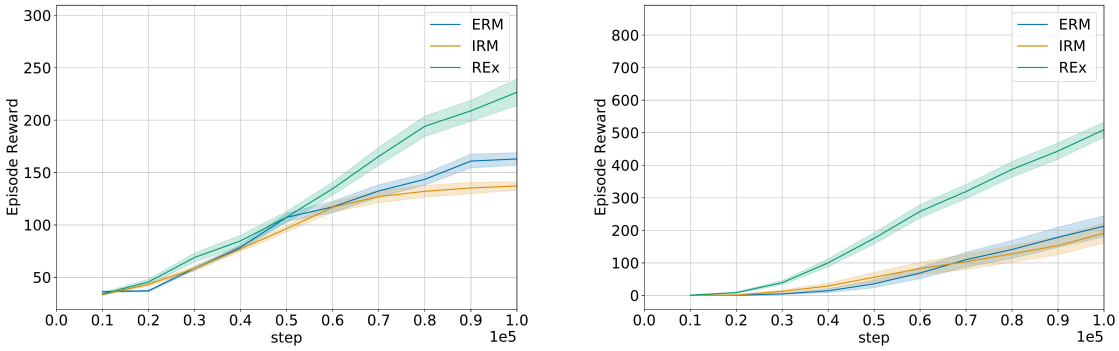


Figure 5.5: Performance and standard error on `walker_walk` (**top**), `finger_spin` (**bottom**).

Algorithm	**ColoredMNIST**	**VLCS**	**PACS**	**OfficeHome**
ERM	52.0 ± 0.1	77.4 ± 0.3	85.7 ± 0.5	67.5 ± 0.5
IRM	51.8 ± 0.1	78.1 ± 0.0	84.4 ± 1.1	66.6 ± 1.0
V-REx	52.1 ± 0.1	77.9 ± 0.5	85.8 ± 0.6	66.7 ± 0.5

Table 5.III: REx, IRM, and ERM all perform comparably on a set of domain generalization benchmarks.

5.4.2 Toy Structural Equation Models (SEMs)

REx's sensitivity to covariate shift can also be a weakness when reallocating capacity towards domains with higher risk does not help the model reduce their risk, e.g. due to irreducible noise. We illustrate this using the linear-Gaussian structural equation model (SEM) tasks introduced by Arjovsky et al. [8]. Like CMNIST, these SEMs include spurious features by construction. They also introduce 1) heteroskedasticity, 2) hidden confounders, and/or 3) elements of X that contain a mixture of causes and effects of Y. These three properties highlight advantages of IRM over ICP [233], as demonstrated empirically by Arjovsky et al. [8]. REx is also able to handle (2) and (3), but it performs poorly in the heteroskedastic tasks. See Appendix 11.1.9.2 for details and Table 11.II for results.

5.4.3 Domain generalization in the DomainBed Suite

Methodologically, it is inappropriate to assume access to the test environment in domain generalization settings, as the goal is to find methods which generalize to *unknown* test distributions. Gulrajani and Lopez-Paz [123] introduced the DomainBed evaluation suite to rigorously compare existing approaches to domain generalization, and found that no method reliably outperformed ERM. We evaluate V-REx on DomainBed using the most commonly used training-domain validation set method for model selection. Due to limited computational resources, we limited ourselves to the 4 cheapest datasets. Results of baseline are taken from Gulrajani and Lopez-Paz [123], who compare with more methods. Results in Table 5.III give the average over 3 different train/valid splits.

5.4.4 Reinforcement learning with partial observability and spurious features

Finally, we turn to reinforcement learning, where covariate shift (potentially favoring REx) and heteroskedasticity (favoring IRM) both occur naturally as a result of randomness in the environment and policy. In order to show the benefits of invariant prediction, we modify tasks from the Deepmind Control Suite [278] to include spurious features in the observation, and train a Soft Actor-Critic [128] agent. REx outperforms both IRM and ERM, suggesting that REx's robustness to covariate shift outweighs the challenges it faces with heteroskedasticity in this setting, see Figure 5.5. We average over 10 runs on `finger_spin` and `walker_walk`, using hyperparameters tuned on `cartpole_swingup` (to avoid overfitting). See Appendix for details and further results.

5.5 Conclusion

We have demonstrated that REx, a method for robust optimization, can provide robustness and hence out-of-distribution generalization in the challenging case where X contains both causes and effects of Y. In particular, like IRM, REx can perform causal identification, but REx can also perform more robustly in the presence of covariate shift. Covariate shift is known to be problematic when models are misspecified, when

training data is limited, or does not cover areas of the test distribution. As such situations are inevitable in practice, REx's ability to outperform IRM in scenarios involving a combination of covariate shift and interventional shift makes it a powerful approach.

CHAPTER 6

DISCUSSION OF "BAYESIAN HYPERNETWORKS"

Author List

David Krueger*
Montreal Institute for Learning Algorithms (MILA);
ElementAI

Chin-Wei Huang*
Montreal Institute for Learning Algorithms (MILA)

Riashat Islam
McGill University

Ryan Turner
Montreal Institute for Learning Algorithms (MILA)

Alexandre Lacoste
ElementAI

Aaron Courville
Montreal Institute for Learning Algorithms (MILA);
CIFAR Fellow

* Equal contributors

6.1 My Contributions

1. Chin-Wei and I independently proposed the core idea of using a normalizing flow to parametrize a variational posterior to Aaron, who put us in touch.

2. I managed the project and did the large majority of the writing, including introduction, related work, and methods.

3. I proposed, implemented, and ran anomaly detection and active learning experiments.

4. I proposed the toy experiment illustrating the capacity of Bayesian hypernetworks to learn a multimodal posterior (Figure 7.2).

6.2 Background

The work presented in this chapter is a **variational inference** method for **Bayesian Deep Learning**. This section provides a brief background on these topics.

In machine learning, we often seek to learn a probabilistic model that approximates the distribution of the data (or some conditional distribution derived from it). However, the quality of this approximation can be difficult to assess, and we may have significant uncertainty as to which of the models under consideration best approximates the underlying data distribution.

At a high-level, the key idea of Bayesian Machine Learning is that we should continue to consider a given model plausible to the extent that the data does not contradict that model.[1] We may also have prior beliefs about which models are more or less likely. Bayesian ML combines such prior beliefs with the likelihood each model assigns to the data to determine how much credence we should place in each of the models, after having seen the training data. This credence is expressed as a distribution and is defined in a mathematically principled way, using Bayes' rule. But note that despite this, Bayesian ML is still a very limited approach to handling uncertainty, since it may require the designer to specify their prior beliefs quite precisely, which is generally infeasible.

Bayes' rule states that:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{6.1}$$

and is a simple consequence of the definition of conditional probability: $P(A|B) \doteq \frac{P(A,B)}{P(B)}$. Nonetheless, this formula has important practical applications and philosophical consequences. **Bayesian updating** refers to updating **prior** beliefs $P(A)$ on the basis of new evidence ($B = b$) to form the **posterior** $P(A|B = b)$. It has been shown to have a number of desirable mathematical and decision theoretic properties, and is often taken as a requirement for rational decision-making.[2]

[1] It is sometimes important to distinguish different classes of models from different values of the parameters of a given model. We elide this distinction in our discussion.

[2] There *are* a number of weaknesses in the arguments for considering Bayesian updating as a normative principle of rational behavior. These are not very widely known, but may have important consequences in

6.2.1 Bayesian Machine Learning

Bayesian machine learning algorithms: 1) express uncertainty over a set of models, defined by their parameters θ, using a prior $P(\theta)$, 2) treat the data D as evidence, and 3) apply Bayesian updating to arrive at a posterior $P(\theta|D)$. Step (3) is known as **Bayesian inference**. This posterior distribution is given by:

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)} \propto P(D|\theta)P(\theta) \tag{6.2}$$

The denominator, $P(D) = \int P(D|\theta)P(\theta)d\theta$ is constant with respect to θ and so can often be ignored.

A predictive distribution for a given test point, x is formed by taking the expected probability with respect to the posterior:

$$P(x|D) = \int P(x|\theta)P(\theta|D)d\theta \tag{6.3}$$

Or, for a conditional distribution:

$$P(y|x,D) = \int P(y|x,\theta)P(\theta|D)d\theta \tag{6.4}$$

Note that we can view these distributions as provided by a probabilistic model defined as a (typically infinite) ensemble of different models – $P(x|\theta)$ or $P(y|x,\theta)$ for every possible value of θ – weighted according to their posterior probability.

The ensemble provided by Bayesian inference should be contrasted with **Maximum Likelihood Estimation (MLE)** of the parameters, which simply seeks to find the single best model $\hat{\theta} = \hat{\theta}_{\text{MLE}} \doteq \text{argmax}_\theta P(D|\theta)$, and make predictions using that model only, i.e. according to $P(x|\hat{\theta})$ or $P(y|x,\hat{\theta})$. In between these two approaches is **Maximum a posteriori (MAP) Estimation**, which is basically MLE with a prior, using $\hat{\theta} = \hat{\theta}_{\text{MAP}} \doteq \text{argmax}_\theta P(D|\theta)P(\theta)$.

Exact Bayesian inference is intractable for DNNs and many other machine learning

AI, and in AI Alignment in particular [40, 71, 102]

models, since it requires computing the likelihood and prior probability for each of the infinitely many possible values of the parameters – even if we were to discretize the parameter space (as happens in practice when implementing models on digital computers), the number of possible parameter values would grow exponentially in the number of parameters. When the prior and likelihood come from specific (pairs of) families of probability distributions (e.g. Gaussian/Gaussian, Beta/Bernoulli, Dirichlet/Multinomial), exact updating *is* possible. But these are very special cases; the likelihood for most reasonably sophisticated machine learning models does not take such a form.

Furthermore, it is often (but not always) intractable to compute the likelihood of a given datapoint under the Bayesian posterior. Instead this may be estimated using finitely many samples from the posterior:

$$P(x|D) = \int P(x|\theta)P(\theta|D)d\theta = \mathbb{E}_{P(\theta|D)}P(x|\theta) \approx \sum_{\theta_i \sim P(\theta|D)} P(x|\theta_i) \tag{6.5}$$

While some methods will provide an approximate posterior in closed form, the main functionality which these methods all provide is the ability to sample from an approximate posterior, $q(\theta) \approx p(\theta|D)$. This can be used to form Monte Carlo estimates of the predictive distribution, $P(y|x,D)$. The difference between $P(y|x,\theta')$ and $P(y|x,\theta'')$ for different values $\theta', \theta'' \sim q(\theta)$ can also provide useful information beyond what is contained in $P(y|x,D)$. In particular, it allows one to distinguish between *reducible* uncertainty about which parameters best model the underlying data distribution, called **epistemic uncertainty**, and *irreducible* uncertainty corresponding to inherent or unmodelable noise, called **aleatoric** uncertainty. This distinction can help guide learning towards reducing epistemic uncertainty, as opposed to, e.g. trying to reduce irreducible noise, which may be impossible and/or lead to overfitting.

While researchers are often concerned with providing theoretical arguments for how methods – even those not originally envisioned as Bayesian, such as dropout [97] – can be viewed as approximations to exact Bayesian inference, *any* method which provides this functionality should be considered as a potential point of comparison for empirical work; two notable examples are ensembles [183] and PriorNets [209].

6.2.2 Approaches to Bayesian Deep Learning

Because it is intractable to compute the posterior of a DNN exactly, Bayesian Deep Learning makes use of a variety of approximate methods. Here we provide a brief overview of several notable general-purpose approaches to approximate Bayesian inference, specifically:

1. Variational Inference
2. Particle-based methods
3. Markov Chain Monte Carlo (MCMC) methods

1) Variational Inference methods approximate the true posterior $P(\theta|D)$ using a parametrized distribution $q_\phi(\theta)$, and update the parameters ϕ of that distribution in order to decrease the KL-divergence. A simple example is **mean-field Variational Inference**, which uses an approximate posterior in which all of the parameters are independent: $q_\phi(\theta) = \prod_i q_{\phi_i}(\theta_i)$. For instance, they might be Gaussians, with $\phi_i = \{\mu_i, \sigma_i\}$, so that $|\phi| = 2|\theta|$. Using a more complicated approximate posterior – such as a full-rank Gaussian, where $|\phi| \approx |\theta|^2$ – is often prohibitively expensive, since DNNs often already use as many parameters as possible given hardware limitations. Bayesian Hypernetworks are a variational inference method, and we describe the mathematics of variational inference and its application in deep learning in Section 6.2.2.

2) Particle-based methods approximate the true posterior using a finite set of discrete "particles", i.e. a mixture of Dirac delta functions. A simple example is training an **ensemble** composed of multiple networks with the same architecture but different training procedure. A prototypical deep ensemble uses different random initializations and different ordering of the training examples as the only source of diversity among ensemble members [183]. Particle-based methods often use more sophisticated training regimes in order to encourage fuller coverage of the Bayesian posterior. In principle, simple ensembling may focus too much on models with high likelihood resulting in too little diversity compared to the true posterior. In practice however, the opposite seems to be true: ensembles appear capable of exploring multiple modes of the posterior in a way that

other approaches fail to [93]. In terms of performance, it is a very strong baseline, often out-performing Bayesian methods, and I consider it the "go-to" method for capturing parameter uncertainty in DNNs. One notable disadvantage of ensembling – and particle-based methods more generally – is that they require training multiple copies of the model, which can be expensive.

3) Markov Chain Monte Carlo (MCMC) methods define a Markov chain whose stationary distribution approximates the true posterior. Bayesian Deep Learning approaches that use MCMC frequently use gradients or gradient estimates to help define the transition probabilities of the Markov chain. For instance, Stochastic Gradient Langevin Dynamics (SGLD) [304] combines Stochastic Gradient Descent (SGD) with appropriately selected Gaussian noise to define the transition operator.

6.3 Contributions of the Work

The primary contribution of this work is the development of a novel and flexible Variational Inference method, called Bayesian Hypernetworks. The basic idea of this method is to parametrize a variational posterior using a generative model – specifically a normalizing flow, so that we can evaluate the likelihood of samples from the variational posterior, in order to estimate the variational lower bound. Ours was one of the first methods of variational inference seeking to learn a highly flexible, multimodal, posterior for Bayesian deep learning. Since our work, many others have introduced approaches for achieving this. An additional contribution is our experimental demonstration that this method can provide qualitative and quantitative benefits over less sophisticated methods of Variational Inference.

CHAPTER 7

BAYESIAN HYPERNETWORKS

ABSTRACT

We study Bayesian hypernetworks: a framework for approximate Bayesian inference in neural networks. A Bayesian hypernetwork h is a neural network which learns to transform a simple noise distribution, $p(\varepsilon) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, to a distribution $q(\theta) := q(h(\varepsilon))$ over the parameters θ of another neural network (the "primary network"). We train q with variational inference, using an invertible h to enable efficient estimation of the variational lower bound on the posterior $p(\theta|\mathscr{D})$ via sampling. In contrast to most methods for Bayesian deep learning, Bayesian hypernets can represent a complex multimodal approximate posterior with correlations between parameters, while enabling cheap iid sampling of $q(\theta)$. We demonstrate these advantages of Bayesian hypernets, which also achieve competitive performance on a suite of tasks that demonstrate the advantage of estimating model uncertainty, including active learning and anomaly detection. In practice, Bayesian hypernets can provide a better defense against adversarial examples than dropout, and also exhibit competitive performance on a suite of tasks which evaluate model uncertainty, including regularization, active learning, and anomaly detection.

7.1 Introduction

Simple and powerful techniques for Bayesian inference of deep neural networks' (DNNs) parameters have the potential to dramatically increase the scope of applications for deep learning techniques. In real-world applications, unanticipated mistakes may be costly and dangerous, whereas anticipating mistakes allows an agent to seek human guidance (as in active learning), engage safe default behavior (such as shutting down), or use a "reject option" in a classification context.

DNNs are typically trained to find the single most likely value of the parameters (the "MAP estimate"), but this approach neglects uncertainty about which parameters are the

best ("parameter uncertainty"), which may translate into higher *predictive* uncertainty when likely parameter values yield highly confident but contradictory predictions. Conversely, Bayesian DNNs model the full posterior distribution of a model's parameters given the data, and thus provides better calibrated confidence estimates, with corresponding safety benefits [4, 97].[1] Maintaining a distribution over parameters is also one of the most effective defenses against adversarial attacks [44].

Techniques for Bayesian DNNs are an active research topic. The most popular approach is variational inference [28, 96], which typically restricts the variational posterior to a simple family of distributions, for instance a factorial Gaussian [28, 121]. Unfortunately, from a safety perspective, variational approximations tend to underestimate uncertainty, by heavily penalizing approximate distributions which place mass in regions where the true posterior has low density. This problem can be exacerbated by using a restricted family of posterior distribution; for instance a unimodal approximate posterior will generally only capture a single mode of the true posterior. With this in mind, we propose learning an extremely flexible and powerful posterior, parametrized by a DNN h, which we refer to as a Bayesian hypernetwork in reference to Ha et al. [125].

A Bayesian hypernetwork (BHN) takes random noise $\varepsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ as input and outputs a sample from the approximate posterior $q(\theta)$ for another DNN of interest (the "primary network"). The key insight for building such a model is the use of an invertible hypernet, which enables Monte Carlo estimation of the entropy term $-\log q(\theta)$ in the variational inference training objective.

We begin the paper by reviewing previous work on Bayesian DNNs, and explaining the necessary components of our approach (Section 7.2). Then we explain how to compose these techniques to yield Bayesian hypernets, as well as design choices which make training BHNs efficient, stable and robust (Section 7.3). Finally, we present experiments which validate the expressivity of BHNs, and demonstrate their competitive performance across several tasks (Section 7.4).

[1] While Bayesian deep learning may capture parameter uncertainty, most approaches, including ours, emphatically do *not* capture uncertainty about which model is correct (e.g., neural net vs decision tree, etc.). Parameter uncertainty is often called "model uncertainty" in the literature, but we prefer our terminology because it emphasizes the existence of further uncertainty about model specification.

7.2 Related work

We begin with an overview of prior work on Bayesian neural networks in Section 7.2.1 before discussing the specific components of our technique in Sections 7.2.2 and 7.2.3.

7.2.1 Bayesian DNNs

Bayesian DNNs have been studied since the 1990s [207, 219]. For a thorough review, see Gal [96]. Broadly speaking, existing methods either 1) use Markov chain Monte Carlo [219, 305], or 2) directly learn an approximate posterior distribution using (stochastic) variational inference [28, 97, 121, 254], expectation propagation [144, 271], or α-divergences [194]. We focus here on the most popular approach: variational inference.

Notable recent work in this area includes Gal and Ghahramani [97], who interprets the popular dropout [223] algorithm as a variational inference method ("MC dropout"). This has the advantages of being simple to implement and allowing cheap samples from $q(\theta)$. Kingma et al. [165] emulates Gaussian dropout, but yields a unimodal approximate posterior, and does not allow arbitrary dependencies between the parameters.

The other important points of reference for our work are Bayes by Backprop (BbB) [28], and multiplicative normalizing flows [203]. Bayes by Backprop can be viewed as a special instance of a Bayesian hypernet, where the hypernetwork only performs an element-wise scale and shift of the input noise (yielding a factorial Gaussian distribution).

More similar is the work of Louizos and Welling [203], who propose and dismiss BHNs due to the issues of scaling BHNs to large primary networks, which we address in Section 7.3.3.[2] Instead, in their work, they use a hypernet to generate scaling factors, $\mathbf{z}$ on the means μ of a factorial Gaussian distribution. Because $\mathbf{z}$ follows a complicated distribution, this forms a highly flexible approximate posterior: $q(\theta) = \int q(\theta|\mathbf{z})q(\mathbf{z})d\mathbf{z}$. However, this approach also requires them to introduce an auxiliary inference network to approximate $q(\mathbf{z}|\theta)$ in order to estimate the entropy term of the variational lower bound,

[2]The idea is also explored by Shi et al. [259], who likewise reject it in favor of their implicit approach which estimates the KL-divergence using a classifier.

resulting in lower bound *on the variational lower bound*.

Finally, the variational autoencoder (VAE) [163, 245] family of generative models is likely the best known application of variational inference in DNNs, but note that the VAE is *not* a Bayesian DNN in our sense. VAEs approximate the posterior over latent variables, given a datapoint; Bayesian DNNs approximate the posterior over model parameters, given a dataset.

7.2.2 Hypernetworks

A hypernetwork [25, 70, 125] is a neural net that outputs parameters of another neural net (the "primary network").[3] The hypernet and primary net together form a single model which is trained by backpropagation. The number of parameters of a DNN scales quadratically in the number of units per layer, meaning naively parametrizing a large primary net requires an impractically large hypernet. One method of addressing this challenge is Conditional Batch Norm (CBN) [81], and the closely related Conditional Instance Normalization (CIN) [150, 291], and Feature-wise Linear Modulation (FiLM) [168, 232], which can be viewed as specific forms of a hypernet. In these works, the weights of the primary net are parametrized directly, and the hypernet only outputs scale (γ) and shift (β) parameters for every neuron; this can be viewed as selecting which features are significant (scaling) or present (shifting). In our work, we employ the related technique of weight normalization [253], which normalizes the input weights for every neuron and introduces a separate parameter g for their scale.

7.2.3 Invertible Generative Models

Our proposed Bayesian hypernetworks employ a differentiable directed generator network (DDGN) [117] as a generative model of the primary net parameters. DDGNs use a neural net to transform simple noise (most commonly isotropic Gaussian) into samples from a complex distribution, and are a common component of modern deep generative

[3] The name "hypernetwork" comes from Ha et al. [125], who describe the general hypernet framework, but applications of this idea in convolutional networks were previously explored by De Brabandere et al. [70] and Bertinetto et al. [25].

models such as variational autoencoders (VAEs) [163, 245] and generative adversarial networks (GANs) [111, 114].

We take advantage of techniques for *invertible* DDGNs developed in several recent works on generative modeling [73, 75] and variational inference of latent variables [166, 243]. Training these models uses the change of variables formula, which involves computing the log-determinant of the inverse Jacobian of the generator network. This computation involves a potentially costly matrix determinant, and these works propose innovative architectures which reduce the cost of this operation but can still express complicated deformations, which are referred to as "normalizing flows".

7.3 Methods

We now describe how variational inference is applied to Bayesian deep nets (Section 7.3.1), and how we compose the methods described in Sections 7.2.2 and 7.2.3 to produce Bayesian hypernets (Section 7.3.2).

7.3.1 Variational Inference

In variational inference, the goal is to maximize a lower bound on the marginal log-likelihood of the data, $\log p(\mathscr{D})$ under some statistical model. This involves both estimating parameters of the model, and approximating the posterior distribution over unobserved random variables (which may themselves also be parameters, e.g., as in the case of Bayesian DNNs). Let $\theta \in \mathbb{R}^D$ be parameters given the Bayesian treatment as random variables, $\mathscr{D}$ a training set of observed data, and $q(\theta)$ a learned approximation to the true posterior $p(\theta|\mathscr{D})$. Since the KL divergence is always non-negative, we have, for *any* $q(\theta)$:

$$\log p(\mathscr{D}) = \mathrm{KL}(q(\theta)\|p(\theta|\mathscr{D})) + \mathbb{E}_q[\log p(\mathscr{D}|\theta) + \log p(\theta) - \log q(\theta)] \tag{7.1}$$

$$\geq \mathbb{E}_q[\log p(\mathscr{D}|\theta) + \log p(\theta) - \log q(\theta)]\,. \tag{7.2}$$

The right hand side of (7.2) is the evidence lower bound, or "ELBO". Since $\log p(\mathscr{D})$ is a constant, an increase in the ELBO yields a corresponding decrease in $\mathrm{KL}(q(\theta)\|p(\theta|\mathscr{D}))$, and thus a better match between the approximate and true posteriors.

The above derivation applies to any statistical model and any dataset. In our experiments, we focus on modeling conditional likelihoods $p(\mathscr{D}) = p(\mathscr{Y}|\mathscr{X})$. Using the conditional independence assumption, we further decompose $\log p(\mathscr{D}|\theta) := \log p(\mathscr{Y}|\mathscr{X},\theta)$ as $\sum_{i=1}^{n} \log p(\mathbf{y}_i|\mathbf{x}_i,\theta)$, and apply stochastic gradient methods for optimization.

7.3.1.0.1 Variational Inference for Deep Networks Computing the expectation in (7.2) is generally intractable for deep nets, but can be estimated by Monte Carlo sampling. For a given value of θ, $\log p(\mathscr{D}|\theta)$ and $\log p(\theta)$ can be computed and differentiated exactly as in a non-Bayesian DNN, allowing training by backpropagation. The entropy term $\mathbb{E}_q[-\log q(\theta)]$ is also straightforward to evaluate for simple families of approximate posteriors such as Gaussians. Similarly, the likelihood of a test data-point under the predictive posterior using S samples can be estimated using Monte Carlo:[4]

$$p(Y=y|X=x,\mathscr{D}) = \int p(Y=y|X=x,\theta)p(\theta|\mathscr{D})d\theta \tag{7.3}$$

$$\approx \frac{1}{S}\sum_{s=1}^{S} p(Y=y|X=x,\theta_s), \quad \theta_s \sim q(\theta)\,. \tag{7.4}$$

7.3.2 Bayesian Hypernets

Bayesian hypernets (BHNs) express a flexible $q(\theta)$ by using a DDGN (section 7.2.3), $h \in \mathbb{R}^D \to \mathbb{R}^D$, to transform random noise $\varepsilon \sim \mathscr{N}(\mathbf{0},\mathbf{I}_D)$ into independent samples from $q(\theta)$. This makes it cheap to compute Monte Carlo estimations of expectations with respect to q; these include the ELBO, and its derivatives, which can be backpropagated to train the hypernet h.

Since BHNs are both trained and evaluated via samples of $q(\theta)$, expressing $q(\theta)$ as a generative model is a natural strategy. However, while DDGNs are convenient to sample

[4]Here we approximate the posterior distribution $p(\theta|\mathscr{D})$ using the approximate posterior $q(\theta)$. We further use S Monte Carlo samples to approximate the integral.

from, computing the entropy term ($\mathbb{E}_q[-\log q(\theta)]$) of the ELBO additionally requires evaluating the likelihood of generated samples, and most popular DDGNs (such as VAEs and GANs) do not provide a convenient way of doing so.[5] In general, these models can be many-to-one mappings, and computing the likelihood of a given parameter value requires integrating over the latent noise variables $\varepsilon \in \mathbb{R}^D$:

$$q(\theta) = \int q(\theta; h(\varepsilon)) q(\varepsilon) d\varepsilon \,. \tag{7.5}$$

To avoid this issue, we use an invertible h, allowing us to compute $q(\theta)$ simply by using the change of variables formula:

$$q(\theta) = q_\varepsilon(\varepsilon) \left| \det \frac{\partial h(\varepsilon)}{\partial \varepsilon} \right|^{-1} , \tag{7.6}$$

where q_ε is the distribution of ε and $\theta = h(\varepsilon)$.

As discussed in Section 7.2.3, a number of techniques have been developed for efficiently training such invertible DDGNs. In this work, we employ both RealNVP (RNVP) [75] and Inverse Autoregressive Flows (IAF) [166]. Note that the latter can be efficiently applied, since we only require the ability to evaluate likelihood of generated samples (not arbitrary points in the range of h, as in generative modeling applications, e.g., Dinh et al. [75]); and this also means that we can use a lower-dimensional ε to generate samples along a submanifold of the entire parameter space, as detailed below.

7.3.3 Efficient parametrization and training of Bayesian hypernets

In order to scale BHNs to large primary networks, we use the weight normalization reparametrization [253] [6]:

$$\theta_j = g\,\mathbf{u}\,, \quad \mathbf{u} := \frac{\mathbf{v}}{\|\mathbf{v}\|_2}\,, \quad g \in \mathbb{R}\,, \tag{7.7}$$

[5] Note that the entropy term is the only thing encouraging dispersion in q; the other two terms of (7.2) encourage the hypernet to ignore the noise inputs ε and deterministically output the MAP-estimate for θ.

[6] Mathematical details can be found in the Appendix, Section 11.2.2.

where θ_j are the input weights associated with a single unit j in the primary network. We only output the scaling factors g from the hypernet, and learn a maximum likelihood estimate of $\mathbf{v}$.[7] This allows us to overcome the computational limitations of naively-parametrized BHNs noted by Louizos and Welling [203], since computation now scales linearly, instead of quadratically, in the number of primary net units. Using this parametrization restricts the family of approximate posteriors, but still allows for a high degree of multimodality and dependence between the parameters.

We also employ weight normalization within the hypernet, and found this stabilizes training dramatically. Initialization plays an important role as well; we recommend initializing the hypernet weights to small values to limit the impact of noise at the beginning of training. We also find clipping the outputs of the softmax to be within $(0.001, 0.999)$ critical for numerical stability.

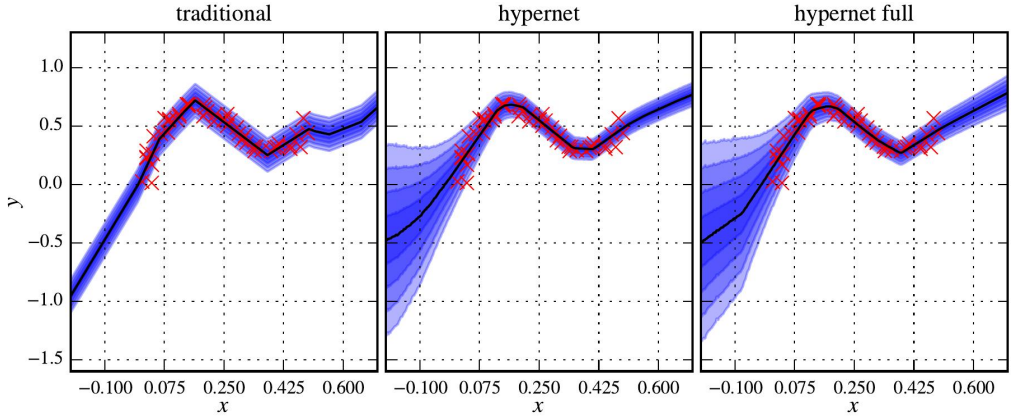


Figure 7.1: Illustration of BHNs (second and third) and a traditional non-Bayesian DNN (first) on the toy problem from Blundell et al. [28]. In the second subplot, we place a prior on the scaling factor g and infer the posterior distribution using a BHN, while in the third subplot the hypernet is used to generate the whole weight matrices of the primary net. Each shaded region represents half a standard deviation in the posterior on the predictive mean. The red crosses are 50 examples from the training dataset.

[7] This parametrization strongly resembles the "correlated" version of variational Gaussian dropout [165, Sec. 3.2]; the only difference is that we restrict the $\mathbf{u}$ to have norm 1.

7.4 Experiments

We perform experiments on MNIST, CIFAR10, and a 1D regression task. There is no single metric for how well a model captures uncertainty; to evaluate our model, we perform experiments on regularization (Section 7.4.2), active learning (Section 7.4.3), anomaly detection (Section 7.4.4), and detection of adversarial examples (Section 7.4.5). Active learning and anomaly detection problems make natural use of uncertainty estimates: In anomaly detection, higher uncertainty indicates a likely anomaly. In active learning, higher uncertainty indicates a greater opportunity for learning. Parameter uncertainty also has regularization benefits: integrating over the posterior creates an implicit ensemble. Intuitively, when the most likely hypothesis predicts "A", but the posterior places more total mass on hypotheses predicting "B", we prefer predicting "B". By improving our estimate of the posterior, we more accurately weigh the evidence for different hypotheses. Adversarial examples are an especially difficult kind of anomaly designed to fool a classifier, and finding effective defenses against adversarial attacks remains an open challenge in deep learning.

For the hypernet architecture, we try both RealNVP [75] and IAF[166] with MADE[105], with 1-layer ReLU-MLP coupling functions with 200 hidden units (each). In general, we find that IAF performs better. We use an isotropic standard normal prior on the scaling factors (g) of the weights of the network. We also use Adam with default hyper-parameter settings [161] and gradient clipping in all of our experiments. Our mini-batch size is 128, and to reduce computation, we use the same noise-sample (and thus the same primary net parameters) for all examples in a mini-batch. We experimented with independent noise, but did not notice any benefit. Our baselines for comparison are Bayes by Backprop (BbB) [28], MC dropout (MCdropout) [97], and non-Bayesian DNN baselines (with and without dropout).

7.4.1 Qualitative results and visualization

We first demonstrate the behavior of the network on the toy 1D-regression problem from Blundell et al. [28] in Figure 7.1. As expected, the uncertainty of the network

increases away from the observed data. We also use this experiment to evaluate the effects of our proposal for scaling BHNs via the weight norm parametrization (Section 7.3.3) by comparing with a model which generates the full set of parameters, and find that the two models produce very similar results, suggesting that our proposed method strikes a good balance between scalability and expressiveness.

Next, we demonstrate the distinctive ability of Bayesian hypernets to learn multi-modal, dependent distributions. Figure 11.12 (appendix) shows that BHNs do learn approximate posteriors with dependence between different parameters, as measured by the Pearson correlation coefficient. Meanwhile, Figure 7.2 shows that BHNs are capable of learning multimodal posteriors. For this experiment, we trained an over-parametrized linear (primary) network: $\hat{y} = a \cdot b \cdot x$ on a dataset generated as $y = x + \varepsilon$, and the BHN learns capture both the modes of $a = b = 1$ and $a = b = -1$.

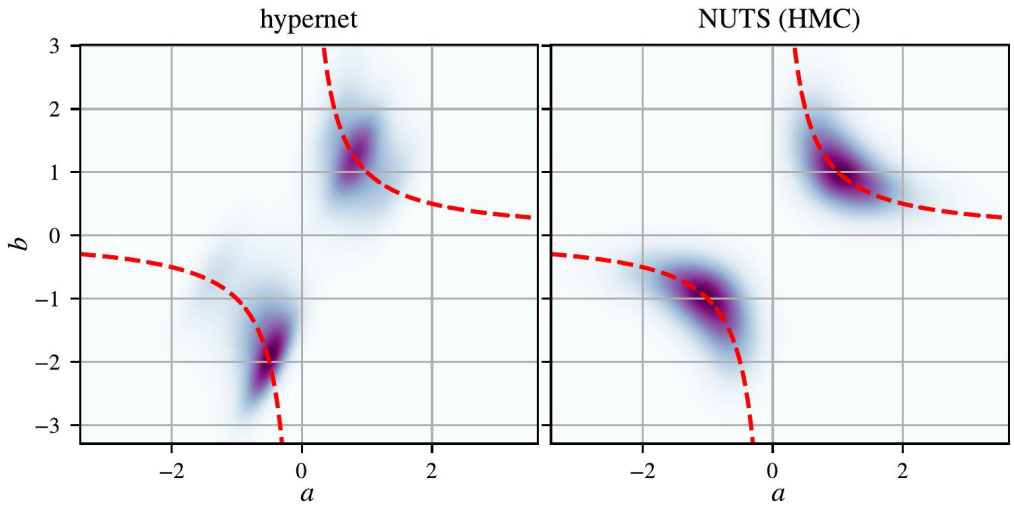


Figure 7.2: Learning the identity function with an overparametrized network: $\hat{y} = a \cdot b \cdot x$. This parametrization results in symmetries shown by the dashed red lines, and the Bayesian hypernetwork assigns significant probability mass to both modes of the posterior ($a = b = 1$ and $a = b = -1$).

7.4.2 Classification

We now show that BHNs act as a regularizer, outperforming dropout and traditional mean field (BbB). Results are presented in Table 7.I. In our experiments, we find that

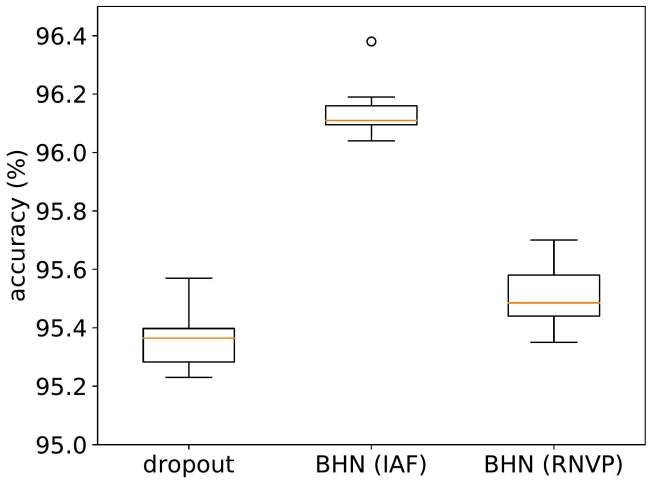


Figure 7.3: Box plot of performance across 10 trials. Bayesian hypernets (BHNs) with inverse autoregressive flows (IAF) consistently outperform the other methods.

BHNs perform on par with dropout on full datasets of MNIST and CIFAR10; furthermore, increasing the flexibility of the posterior by adding more coupling layers improves performance, especially compared with models with 0 coupling layers, which cannot model dependencies between the parameters. We also evaluate on a subset of MNIST (the first 5,000 examples); results are presented in the last two columns of Table 7.I. Replicating these experiments (with 8 coupling layers) for 10 trials yields Figure 7.3.

In these MNIST experiments, we use MLPs with 2 hidden layers of 800 or 1200 hidden units each. For CIFAR10, we train a convolutional neural net (CNN) with 4 hidden layers of $[64, 64, 128, 128]$ channels, 2×2 max pooling after the second and the fourth layers, filter size of 3, and a single fully connected layer of 512 units.

7.4.3 Active Learning

We now turn to active learning, where we compare to the MNIST experiments of Gal et al. [99], replicating their architecture and training procedure. Briefly, they use an initial dataset of 20 examples (2 from each class), and acquire 10 new examples at a time, training for 50 epochs between each acquisition. While Gal et al. [99] re-initialize the network after every acquisition, we found that "warm-starting" from the current learned

Table 7.I: Generalization results on MNIST and CIFAR10 for BHNs with different numbers of RealNVP coupling layers (#), and comparison methods (dropout / maximum likelihood (MLE)). Bayes-by-backprop [28] (*) models each parameter as an independent Gaussian, which is equivalent to using a hypernet with 0 coupling layers. We achieved a better result outputting a distribution over scaling factors (only). MNIST 5000 (A) and (B) are generalization results on subset (5,000 training data) of MNIST, (A) MLP with 800 hidden nodes. (B) MLP with 1,200 hidden nodes.

MNIST 50,000		CIFAR10 50,000		MNIST 5,000 (A)		MNIST 5,000 (B)	
#	Accuracy	#	Accuracy	#	Accuracy	#	Accuracy
0	98.28% (98.01%*)	0	67.83%	0	92.06%	0	90.91%
2	98.39%	4	74.77%	8	94.25%	8	96.27%
4	98.47%	8	**74.90%**	12	**96.16%**	12	**96.51%**
6	98.59%	dropout	74.08%	dropout	95.58%	dropout	95.52%
8	98.63%	MLE	72.75%				
dropout	**98.73%**						

parameters was essential for good performance with BHNs, although it is likely that longer training or better initialization schemes could perform the same role. Overall, warm-started BHNs suffered at the beginning of training, but outperformed all other methods for moderate to large numbers of acquisitions.

7.4.4 Anomaly Detection

Table 7.II: Anomaly detection on MNIST. Since we use the same datasets as Hendrycks and Gimpel [138], we have the same base error rates, and refer the reader to that work.

Dataset	MLP			MC dropout			BHN		
	ROC	PR(+)	PR(−)	ROC	PR(+)	PR(−)	ROC	PR(+)	PR(−)
Uniform	96.99	97.99	94.71	98.90	99.15	98.63	98.97	99.27	98.52
OmniGlot	94.92	95.63	93.85	95.87	96.44	94.84	94.89	95.56	93.64
CIFARbw	95.55	96.47	93.72	98.70	98.98	98.39	96.63	97.25	95.78
Gaussian	87.70	87.66	88.05	97.70	98.11	96.94	89.22	86.62	89.85
notMNIST	81.12	97.56	39.70	97.78	99.78	78.53	90.07	98.51	56.59

For anomaly detection, we take Hendrycks and Gimpel [138] as a starting point, and

[9]For the deterministic baseline, the value of the BALD acquisition function is always zero, and so acquisitions should be random, but due to numerical instability this is not the case in our implementation; surprisingly, we found the BALD values our implementation computes provide a better-than-random acquisition function (compare the blue line in the top and bottom plots).

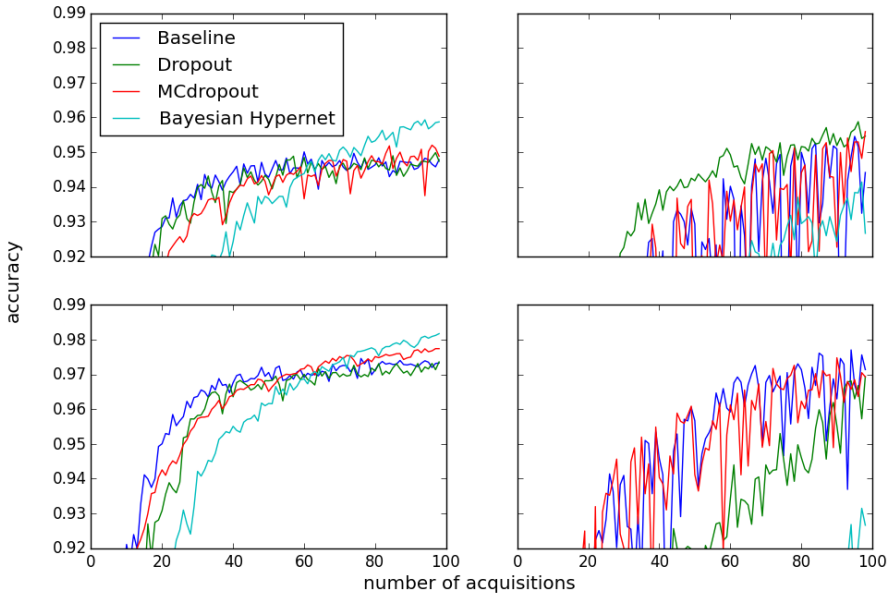


Figure 7.4: Active learning: Bayesian hypernets outperform other approaches after sufficient acquisitions when warm-starting (left), for both random acquisition function (top) and BALD acquisition function (bottom). Warm-starting improves stability for all methods, but hurts performance for other approaches, compared with randomly re-initializing parameters as in Gal et al. [99] (right). We also note that the baseline model (no dropout) is competitive with MCdropout, and outperforms the Dropout baseline used by [99].[9] These curves are the average of three experiments.

perform the same suite of MNIST experiments, evaluating the ability of networks to determine whether an input came from their training distribution ("Out of distribution detection"). Hendrycks and Gimpel [138] found that the confidence expressed in the softmax probabilities of a (non-Bayesian) DNN trained on a single dataset provides a good signal for both of these detection problems. We demonstrate that Bayesian DNNs outperform their non-Bayesian counterparts.

Just as in active learning, in anomaly detection, we use MC to estimate the predictive posterior, and use this to score datapoints. For active learning, we would generally like to acquire points where there is higher uncertainty. In a well-calibrated model, these points are also likely to be challenging or anomalous examples, and thus acquisition functions from the active learning literature are good candidates for scoring anomalies.

We consider all of the acquisition functions listed in [99] as possible scores for the Area Under the Curve (AUC) of Precision-Recall (PR) and Receiver Operating Characteristic (ROC) metrics, but found that the maximum confidence of the softmax probabilities (i.e., "variation ratio") acquisition function used by Hendrycks and Gimpel [138] gave the best performance. Both BHN and MCdropout achieve significant performance gains over the non-Bayesian baseline, and MCdropout performs significantly better than BHN in this task. Results are presented in Table 7.II.

Second, we follow the same experimental setup, using all the acquisition functions, and exclude one class in the training set of MNIST at a time. We take the excluded class of the training data as out-of-distribution samples. The result is presented in Table 11.VII (Appendix). This experiment shows the benefit of using scores that reflect dispersion in the posterior samples (such as mean standard deviation and BALD value) in Bayesian DNNs.

7.4.5 Adversary Detection

Finally, we consider this same anomaly detection procedure as a novel tool for detecting adversarial examples. Our setup is similar to Li and Gal [194] and Louizos and Welling [203], where it is shown that when more perturbation is added to the data, model uncertainty increases and then drops. We use the Fast Gradient Sign method (FGS) [115] for adversarial attack, and use one sample of our model to estimate the gradient.[10] We find that, compared with dropout, BHNs are less confident on data points which are far from the data manifold. In particular, BHNs constructed with IAF consistently outperform RealNVP-BHNs and dropout in detecting adversarial examples and errors. Results are shown in Figure 7.5.

[10] Li and Gal [194] and Louizos and Welling [203] used 10 and 1 model samples, respectively, to estimate gradient. We report the result with 1 sample; results with more samples are given in the appendix.

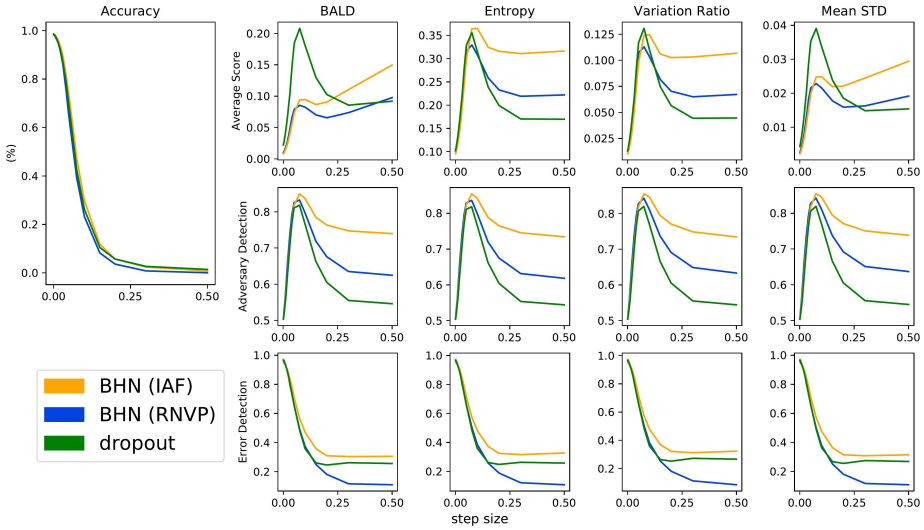


Figure 7.5: Adversary detection: Horizontal axis is the step size of the FGS algorithm. While accuracy drops when more perturbation is added to the data (left), uncertainty measures also increase (first row). In particular, the BALD and Mean STD scores, which measure epistemic uncertainty, are strongly increasing for BHNs, but *not* for dropout. The second row and third row plots show results for adversary detection and error detection (respectively) in terms of the AUC of ROC (y-axis) with increasing perturbation along the x-axis. Gradient direction is estimated with one Monte Carlo sample of the weights/dropout mask.

7.5 Conclusions

We introduce Bayesian hypernets (BHNs), a new method for variational Bayesian deep learning which uses an invertible hypernetwork as a generative model of parameters. BHNs feature efficient training and sampling, and can express complicated multimodal distributions, thereby addressing issues of overconfidence present in simpler variational approximations. We present a method of parametrizing BHNs which allows them to scale successfully to real world tasks, and show that BHNs can offer significant benefits over simpler methods for Bayesian deep learning. Future work could explore other methods of parametrizing BHNs, for instance using the same hypernet to output different subsets of the primary net parameters.

CHAPTER 8

DISCUSSION OF "NEURAL AUTOREGRESSIVE FLOWS"

Author List

Chin-Wei Huang*
MILA;
University of Montreal;
Element AI

David Krueger*
MILA;
University of Montreal;

Element AI

Alexandre Lacoste
Element AI

Aaron Courville
MILA;
University of Montreal;
CIFAR fellow

* Equal contributors

8.1 My Contributions

1. Chin-Wei raised the issue of improving multimodality of normalizing flow models, and we had discussed and tried several approaches, including some that I proposed.

2. Chin-Wei came up with the DSF/DDSF models.

3. I came up with the name "neural autoregressive flows", and the general framing (in terms of conditioner and transformer), including the general "recipe" (i.e. requirements) for constructing a NAF.

4. I mostly wrote the abstract, intro, background, and methods sections, and helped write and edit the rest of the paper.

5. I ran most of the density estimation experiments.

6. I helped check the proofs, explain the steps informally, and tie up a few loose ends in them.

8.2 Background

The work presented in this chapter provides a new approach towards building **normalizing flows** making use of deep **autoregressive models**. This section provides a brief background on these topics.

8.2.1 Autoregressive models

Autoregressive models are an approach to unsupervised learning for time-series data, where the value at every time-step is predicted based on values of previous time-steps (and a noise term ε):

$$P(x_t|x_{t-1},...,x_{t-p}) = f_\theta(x_{t-1},...,x_{t-p},\varepsilon) \tag{8.1}$$

In classical statistics, autoregressive models assume the relationship between previous and current time-steps is linear. In contrast, *deep* autoregressive models make these predictions using neural networks, and are also commonly applied to other types of sequentially structured data.[1] For instance, autoregressive models have achieved state-of-the-art results on image data when treating images as a sequence of pixels or image patches, e.g. by concatenating rows or columns [80, 294].

A classic deep learning approach to autoregressive modeling is to use some variety of **Recurrent Neural Network (RNN)** [86], such as a **Long Short-Term Memory (LSTM)** RNN [40]. RNNs maintain a "hidden state" which is a learned function of all previously observed time-steps; thus, in principle, they have an arbitrarily long memory.

Recent works, however, have found better performance using other architectures – such as **convolutional neural networks (CNNs)** [185], **transformers**[2] [298], or some combination [52] – that use a limited "context" of the previous p time-steps; in statistics parlance, p is the **order** of the autoregressive model.

[1] They can also be applied to data without any inherent sequential structure by choosing an (e.g. arbitrary) ordering of the data.

[2] Note that "transformer" refers to an entirely different concept in the context of Neural Autoregressive Flows.

Deep autoregressive models are typically capable of both: 1) **evaluating** the probability of an arbitrary candidate sequence, and 2) **sampling** novel sequences from the distribution learned by the model. Models are commonly trained using maximum likelihood estimation (MLE), which only requires the first operation. This gives CNNs and Transformers a significant computational advantage over RNNs at training time. While all 3 models require serial computation at successive time-steps in order to *sample*, only RNNs require serial computation for evaluating the likelihood; CNNs and transformers can evaluate likelihood at every time-step in parallel.

8.2.2 Normalizing Flows

Normalizing Flows refer to a family of techniques for estimating local changes in probability density that result from applying invertible transformations to samples from a probability distribution. For instance, samples from a normal distribution $z \sim \mathcal{N}([0,1])$ might be transformed into samples from a more complicated distribution $p_x(x)$ by passing them through the inverse cumulative distribution function (CDF) of p, a method known as **inverse transform sampling**. A Normalizing Flow is a model that parametrizes and learns such a transformation, e.g. in order to approximately sample some target distribution. Importantly, the density of a given point does in fact change as a result of such transformations, in accordance with the change-of-variables formula of calculus:

$$p_x(x) = \left| \frac{\partial f(z)}{\partial z} \right|^{-1} p_z(z) \tag{8.2}$$

To understand intuitively why the density changes, consider the (invertible) mapping $z \to 2z$ applied to samples from $U(0,1)$. Applying this mapping stretches the same total probability mass (of 1) across the interval $(0,2)$, meaning the density *must* decrease proportionally. The area under the PDF in this case is a rectangle, whose total area must be 1, so making the horizontal side length 2 means the height becomes 1/2. More generally, an invertible mapping may stretch or compress different areas of space differently, and the change-of-variables formula accounts for such differences precisely.

To see the advantage of using *invertible* mappings in particular, we can compare with

approaches without such a restriction. But first, for concreteness, let's consider a specific application scenario for transforming probability distributions: generative modeling and density estimation.

Normalizing flows are probably best known as generative models of images [74, 77, 162]. In this application, they transform random noise (e.g. i.i.d. samples from $\mathcal{N}([0,1])$) into samples resembling images such as photographs of human faces.

In this context, maximum likelihood training requires computing the probability the model assigns to real images from the training set. Examining the change-of-variables formula, we see that the log-likelihood has two terms, which can be computed in closed form:

1. The log-likelihood of $z = f^{-1}(x)$. The probability of x is higher when the probability of the corresponding noise variable is higher under $p_z(z)$.
2. The log-determinant of Jacobian, $\log \left| \frac{\partial f(z)}{\partial z} \right|^{-1}$. The probability of x is higher when f has a small derivative; intuitively, this means f is locally *contracting* around x, i.e. pushing more mass into a smaller volume, and thus increasing the probability density at x.

Now, let us return to contrasting with a non-invertible model. Specifically, let's assume that we have a generative model which assigns non-zero probability to every possible image, but which may map different noise samples to the same image. In this case, when we want to estimate the probability of an image, we need to perform an integral over all of the noise samples in the pre-image of that image. This is typically intractable, although it can be estimated reasonably well in some cases, e.g. via importance sampling in Variational Autoencoders [42, 246].

Generative modelling is only one application of Normalizing Flows, others include density estimation [230], variational inference [167], maximum entropy problems [160], and hierarchical reinforcement learning [127]. Broadly speaking, Normalizing Flows are an incredibly flexible tool for probabilistic model – they can potentially be inserted anywhere where samples from a distribution are being passed between components of a model. Unlike many other types of transformation (e.g. typical DNNs), many

Normalizing Flows are algorithmically "invisible" in the sense that if the probability of the inputs/outputs is known, then so is the probability of the outputs/inputs; the same is true of gradients of the probability/likelihood. It is important to note, however, that many Normalizing Flow models are "directional": while they define an invertible mapping, computing this mapping and/or estimating the change in density it induces may only be tractable or efficient in one direction (similar to the issue with RNNs discussed in Section 8.2.1).

I'll conclude this section with giving a simple example of a Normalizing Flow for concreteness. This will also illustrate the role that careful architecture design plays in 1) guaranteeing invertibility and 2) enabling efficient computation/estimation of the change-of-variables formula. Our example Normalizing Flow will be a standard Multi-Layer Perceptron (MLP), which can easily be made invertible by using invertible activation functions (such as leaky ReLUs [205]) and square (invertible) weight matrices. Computing the change of variables formula for such a model is expensive, however, requiring computation which is roughly cubic in the input dimensionality to invert the weight matrices. However, for upper/lower triangular weight matrices, the determinant of the inverse can be computed based only on the diagonal terms. In fact, affine autoregressive flow models use a very similar approach, which guarantees that the inverse Jacobian is triangular while allowing richer dependencies between x_i and $x_{1,\dots,i-1}$.

8.2.3 Autoregressive Models and Normalizing Flows

In a precursor to our work, Papamakarios et al. [230] observed an equivalence between (certain) autoregressive models and normalizing flows. Deep autoregressive models often model $P(x_t|x_{t-1},\dots)$ as a Gaussian, with parameters $\mu_t(x_{t-1},\dots)$ and $\sigma_t^2(x_{t-1},\dots)$ output from a neural network. These parameters define an invertible transformation between possible values of x_t and the value of the corresponding noise, ε_t, specifically, we have the following affine relationship:

$$x_t = \mu_t + \sigma_t * \varepsilon_t \tag{8.3}$$

or, equivalently:

$$\varepsilon_t = \frac{(x_t - \mu_t)}{\sigma_t}. \tag{8.4}$$

Thus, considering all t, we can view this model as an invertible mapping from $\varepsilon_1, ..., \varepsilon_T$ to $x_1, ..., x_T$, i.e. a normalizing flow. Furthermore, the Jacobian matrix for this mapping is triangular, and so its determinant is given simply by the product of its diagonal entries.

This same trick with the Jacobian is used in the work presented in this chapter. In fact, the core idea of the work is simple, in light of this relationship with autoregressive models. The transformation of ε_t into x_t can is replaced by a more expressive invertible transformation, and the parameters for that transformation can be produced as outputs of an autoregressive model in exactly the same fashion as just described.

8.3 Contributions of the Work

This paper advanced the field of normalizing flows with: 1) state-of-the-art empirical results, 2) a qualitative difference in the ability to model multi-modal target distributions, 3) the first model with a proven capacity to approximate any (continuous) probability density to arbitrary precision. As discussed earlier, one application where the expressivity of such a method may be important is modelling the (complicated and multi-modal) Bayesian posterior of a neural network's parameters.Normalizing flows are an increasingly popular family of generative models, and this work outlined a general recipe for constructing them, and inspired a number of follow-up works. It has also become somewhat popular to discuss universal approximation results.

It has become somewhat popular to discuss universal approximation results. Autoregressive flows have been proven to *not* be universal approximators. However, there doesn't seem to have been much work on actually trying to move beyond such results to give more practical guarantees or guidance e.g. How much capacity do you need? How hard is optimization?

CHAPTER 9

NEURAL AUTOREGRESSIVE FLOWS

ABSTRACT

Normalizing flows and autoregressive models have been successfully combined to produce state-of-the-art results in density estimation, via Masked Autoregressive Flows (MAF) [230], and to accelerate state-of-the-art WaveNet-based speech synthesis to 20x faster than real-time [295], via Inverse Autoregressive Flows (IAF) [167]. We unify and generalize these approaches, replacing the (conditionally) affine univariate transformations of MAF/IAF with a more general class of invertible univariate transformations expressed as monotonic neural networks. We demonstrate that the proposed **neural autoregressive flows (NAF)** are universal approximators for continuous probability distributions, and their greater expressivity allows them to better capture multimodal target distributions. Experimentally, NAF yields state-of-the-art performance on a suite of density estimation tasks and outperforms IAF in variational autoencoders trained on binarized MNIST. [1]

9.1 Introduction

Invertible transformations with a tractable Jacobian, also known as **normalizing flows**, are useful tools in many machine learning problems, for example: (1) In the context of **deep generative models**, training necessitates evaluating data samples under the model's inverse transformation [76]. Tractable density is an appealing property for these models, since it allows the objective of interest to be directly optimized; whereas other mainstream methods rely on alternative losses, in the case of intractable density models [164, 246], or implicit losses, in the case of adversarial models [113]. (2) In the context of **variational inference** [244], they can be used to improve the variational approximation to the posterior by parameterizing more complex distributions. This is important since a poor variational approximation to the posterior can fail to reflect the

[1] Implementation can be found at https://github.com/CW-Huang/NAF/

right amount of *uncertainty*, and/or be biased [289], resulting in inaccurate and unreliable predictions. We are thus interested in improving techniques for normalizing flows.

Recent work by Kingma et al. [167] reinterprets autoregressive models as invertible transformations suitable for constructing normalizing flows. The inverse transformation process, unlike sampling from the autoregressive model, is not sequential and thus can be accelerated via parallel computation. This allows multiple layers of transformations to be stacked, increasing expressiveness for better variational inference [167] or better density estimation for generative models [230]. Stacking also makes it possible to improve on the sequential conditional factorization assumed by autoregressive models such as PixelRNN or PixelCNN [294], and thus define a more flexible joint probability.

We note that the normalizing flow introduced by Kingma et al. [167] only applies an affine transformation of each scalar random variable. Although this transformation is conditioned on preceding variables, the resulting flow can still be susceptible to bad local minima, and thus failure to capture the multimodal shape of a target density; see Figure 9.1 and 9.2.

9.1.1 Contributions of this work

We propose replacing the conditional affine transformation of Kingma et al. [167] with a more rich family of transformations, and note the requirements for doing so. We determine that very general transformations, for instance parametrized by deep neural networks, are possible. We then propose and evaluate several specific monotonic neural network architectures which are more suited for learning multimodal distributions. Concretely, our method amounts to using an autoregressive model to output the weights of multiple independent transformer networks, each of which operates on a single random variable, replacing the affine transformations of previous works.

Empirically, we show that our method works better than the state-of-the-art affine autoregressive flows of Kingma et al. [167] and Papamakarios et al. [230], both as a sample generator which captures multimodal target densities with higher fidelity, and as a density model which more accurately evaluates the likelihood of data samples drawn from an unknown distribution.

We also demonstrate that our method is a universal approximator on proper distributions in real space, which guarantees the expressiveness of the chosen parameterization and supports our empirical findings.

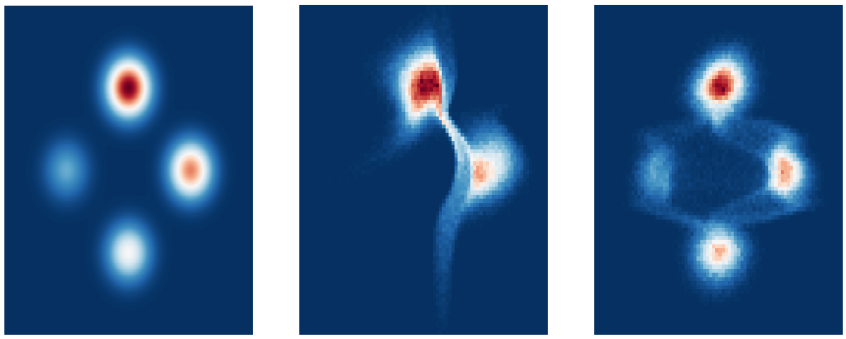


Figure 9.1: Energy function fitting using IAF.
Left: true distribution. Center: IAF-affine. Right: IAF-DSF.

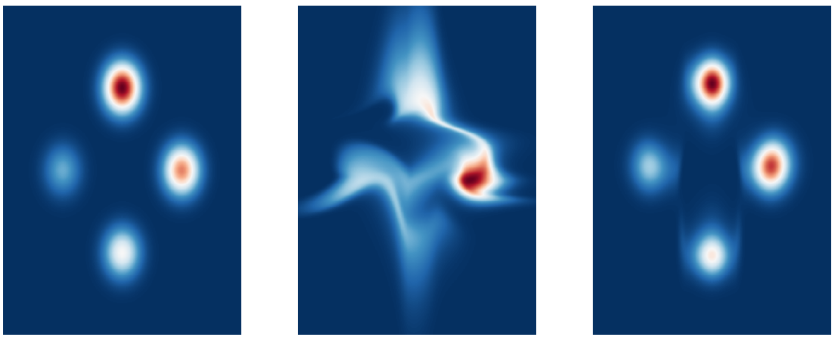


Figure 9.2: Density estimation using MAF.
Left: true distribution. Center: MAF-affine. Right: MAF-DSF.

9.2 Background

A (finite) **normalizing flow (NF)**, or **flow**, is an invertible function $f_\theta : \mathscr{X} \to \mathscr{Y}$ used to express a transformation between random variables [2]. Since f is invertible, the change

[2] We use **x** and **y** to denote inputs and outputs of a function, *not* the inputs and targets of a supervised learning problem.

of variables formula can be used to translate between densities $p_Y(\mathbf{y})$ and $p_X(\mathbf{x})$:

$$p_Y(\mathbf{y}) = \left| \frac{\partial f(\mathbf{x})}{\partial \mathbf{x}} \right|^{-1} p_X(\mathbf{x}) \tag{9.1}$$

The determinant of f's Jacobian appears on the right hand side to account for the way in which f can (locally) expand or contract regions of X, thereby lowering or raising the resulting density in those regions' images in Y. Since the composition of invertible functions is itself invertible, complex NFs are often formed via function composition (or "stacking") of simpler NFs.

Normalizing flows are most commonly trained to produce an output distribution $p_Y(\mathbf{y})$ which matches a target distribution (or, more generally, energy function) $p_{\text{target}}(\mathbf{y})$ as measured by the KL-divergence $KL(p_Y(\mathbf{y})||p_{\text{target}}(\mathbf{y}))$. When X or Y is distributed by some simple distribution, such as uniform or standard normal, we call it an unstructured noise; and we call it a structured noise when the distribution is complex and correlated. Two common settings are maximum likelihood and variational inference. Note that these two settings are typically viewed as optimizing different directions of the KL-divergence, whereas we provide a unified view in terms of different input and target distributions. A detailed derivation is presented in the appendix.

For maximum likelihood applications [76, 230], $p_{\text{target}}(\mathbf{y})$ is typically a simple prior over latent variable $\mathbf{y}$, and f attempts to disentangle the complex empirical distribution of the data, $p_X(\mathbf{x})$ into a simple latent representation $p_Y(\mathbf{y})$ matching the prior (*structured to unstructured*) [3].

In a typical application of variational inference [167, 244], $p_{\text{target}}(\mathbf{y})$ is a complex posterior over latent variables $\mathbf{y}$, and f transforms a simple input distribution (for instance a standard normal distribution) over $\mathbf{x}$ into a complex approximate posterior $p_Y(\mathbf{y})$ (*u*nstructured to structured). In either case, since p_X does not depend on θ, the gradients

[3] It may also be possible to form a generative model from such a flow, by passing samples from the prior $p_{\text{target}}(\mathbf{y})$ through f^{-1}, although the cost of doing so may vary. For example, RealNVP [76] was devised as a generative model, and its inverse computation is as cheap as its forward computation, whereas MAF [230] is designed for density estimation and is much more expensive to sample from. For the NAF architectures we employ, we do not have an analytic expression for f^{-1}, but it is possible to approximate it numerically.

of the KL-divergence are typically estimated by Monte Carlo:

$$\begin{aligned}\nabla_\theta \mathscr{D}_{KL}\big(p_Y(\mathbf{y})||p_{\text{target}}(\mathbf{y})\big) \\ &= \nabla_\theta \int_{\mathscr{Y}} p_Y(\mathbf{y}) \log \frac{p_Y(\mathbf{y})}{p_{\text{target}}(\mathbf{y})} \mathrm{d}\mathbf{y} \\ &= \int_{\mathscr{X}} p_X(\mathbf{x}) \nabla_\theta \log \frac{p_Y(\mathbf{y})}{p_{\text{target}}(\mathbf{y})} \mathrm{d}\mathbf{x} \end{aligned} \tag{9.2}$$

Applying the change of variables formula from Equation 1 to the right hand side of Equation 2 yields:

$$\mathbb{E}_{\substack{\mathbf{x}\sim p_X(\mathbf{x}) \\ \mathbf{y}=f_\theta(\mathbf{x})}} \left[\nabla_\theta \log \left| \frac{\partial f_\theta(\mathbf{x})}{\partial \mathbf{x}} \right|^{-1} p_X(\mathbf{x}) - \nabla_\theta \log p_{\text{target}}(\mathbf{y}) \right] \tag{9.3}$$

Thus for efficient training, the following operations must be tractable and cheap:

1. Sampling $\mathbf{x} \sim p_X(\mathbf{x})$
2. Computing $\mathbf{y} = f(\mathbf{x})$
3. Computing the gradient of the log-likelihood of $\mathbf{y} = f(\mathbf{x})$; $\mathbf{x} \sim p_X(\mathbf{x})$ under both $p_Y(\mathbf{y})$ and $p_{\text{target}}(\mathbf{y})$
4. Computing the gradient of the log-determinant of the Jacobian of f

Research on constructing NFs, such as our work, focuses on finding ways to parametrize flows which meet the above requirements while being maximally flexible in terms of the transformations which they can represent. Note that some of the terms of of Equation 3 may be constant with respect to θ [4] and thus trivial to differentiate, such as $p_X(\mathbf{x})$ in the maximum likelihood setting.

Affine autoregressive flows (AAFs) [5], such as inverse autoregressive flows (IAF) [167], are one particularly successful pre-existing approach. Affine autoregressive flows

[4] There might be some other parameters other than θ that are learnable, such as parameters of p_X and p_{target} in the variational inference and maximum likelihood settings, respectively.

[5] Our terminology differs from previous works, and hence holds the potential for confusion, but we believe it is apt. Under our unifying perspective, NAF, IAF, AF, and MAF all make use of the same principle, which is an invertible transformer conditioned on the outputs of an autoregressive (and emphatically *not* an *inverse* autoregressive) conditioner.

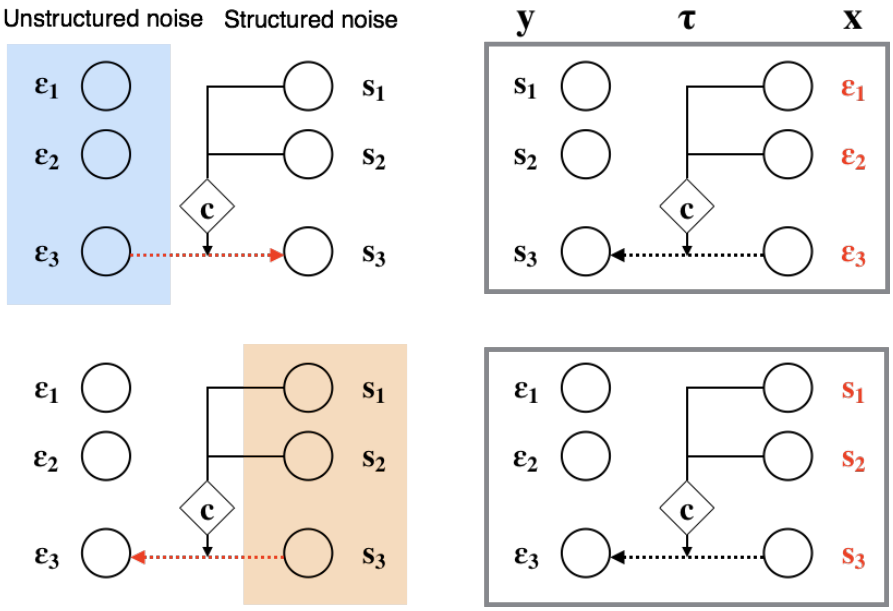


Figure 9.3: Difference between autoregressive and inverse autoregressive transformations (left), and between IAF and MAF (right). **Upper left**: sample generation of an autoregressive model. Unstructured noise is transformed into structured noise. **Lower left**: inverse autoregressive transformation of structured data. Structured variables are transformed into unstructured variables. **Upper right**: IAF-style sampling. **Lower right**: MAF-style evaluation of structured data. ε represents unstructured noise and s represents structured noise.

yield a triangular Jacobian matrix, so that the log-determinant can be computed in linear time, as the sum of the diagonal entries on log scale. In AAFs, the components of $\mathbf{x}$ and $\mathbf{y}$ are given an order (which may be chosen arbitrarily), and y_t is computed as a function of $x_{1:t}$. Specifically, this function can be decomposed via an autoregressive **conditioner**, c, and an invertible **transformer**, τ, as [6]:

$$y_t \doteq f(x_{1:t}) = \tau(c(x_{1:t-1}), x_t) \tag{9.4}$$

It is possible to efficiently compute the output of c for all t in a single forward pass using a model such as **MADE** [105], as pointed out by Kingma et al. [167].

[6] Dinh et al. [73] use m and g^{-1} to denote c and τ, and refer to them as the "coupling function" and "coupling law", respectively.

In previous work, τ is taken to be an affine transformation with parameters $\mu \in \mathbb{R}, \sigma > 0$ output from c. For instance Dinh et al. [76] use:

$$\tau(\mu, \sigma, x_t) = \mu + \sigma x_t \tag{9.5}$$

with σ produced by an exponential nonlinearity. Kingma et al. [167] use:

$$\tau(\mu, \sigma, x_t) = \sigma x_t + (1 - \sigma)\mu \tag{9.6}$$

with σ produced by a sigmoid nonlinearity. Such transformers are trivially invertible, but their relative simplicity also means that the expressivity of f comes entirely from the complexity of c and from stacking multiple AAFs (potentially using different orderings of the variables) [7]. However, the only requirements on τ are:

1. The transformer τ must be invertible as a function of x_t.
2. $\frac{dy_t}{dx_t}$ must be cheap to compute.

This raises the possibility of using a more powerful transformer in order to increase the expressivity of the flow.

9.3 Neural autoregressive flows

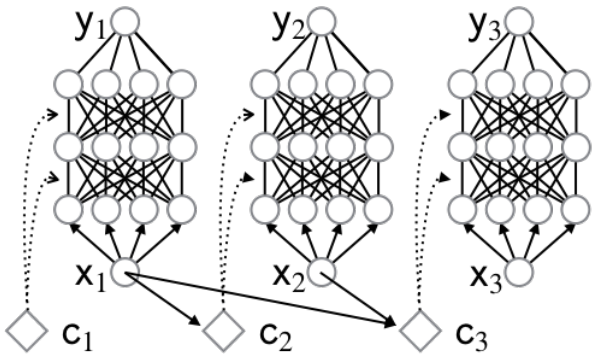
We propose replacing the affine transformer used in previous works with a neural network, yielding a more rich family of distributions with only a minor increase in computation and memory requirements. Specifically,

$$\tau(c(x_{1:t-1}), x_t) = \text{DNN}(x_t;\ \phi = c(x_{1:t-1})) \tag{9.7}$$

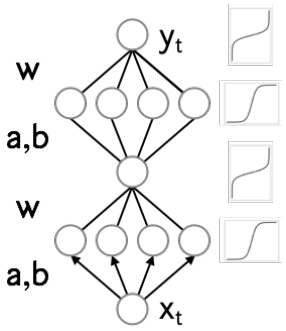
is a deep neural network which takes the scalar x_t as input and produces y_t as output, and its weights and biases are given by the outputs of $c(x_{1:t-1})$[8] (see Figure 9.4(a)). We refer

[7] Permuting the order of variables is itself a normalizing flow that does not expand or contract space and can be inverted by another permutation.

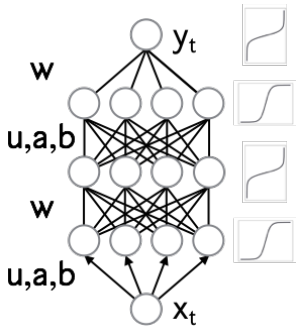
[8] We'll sometimes write τ_c for $\tau(c(x_{1:t-1}), \cdot)$.



(a) Neural autoregressive flows (NAF)



(b) DSF



(c) DDSF

Figure 9.4: **Top:** In neural autoregressive flows, the transformation of the current input variable is performed by an MLP whose parameters are output from an autoregressive conditioner model, $c_t \doteq c(x_{1:t-1})$, which incorporates information from previous input variables. **Bottom:** The architectures we use in this work: deep sigmoidal flows (DSF) and deep dense sigmoidal flows (DDSF). See section 9.3.1 for details.

to these values ϕ as **pseudo-parameters**, in order to distinguish them from the statistical parameters of the model.

We now state the condition for NAF to be strictly monotonic, and thus invertible (as per requirement 1):

Proposition 1. *Using strictly positive weights and strictly monotonic activation functions for τ_c is sufficient for the entire network to be strictly monotonic.*

Meanwhile, $\frac{dy_t}{dx_t}$ and gradients wrt the pseudo-parameters [9] can all be computed

[9] Gradients for pseudo-parameters are backpropagated through the conditioner, c, in order to train its parameters.

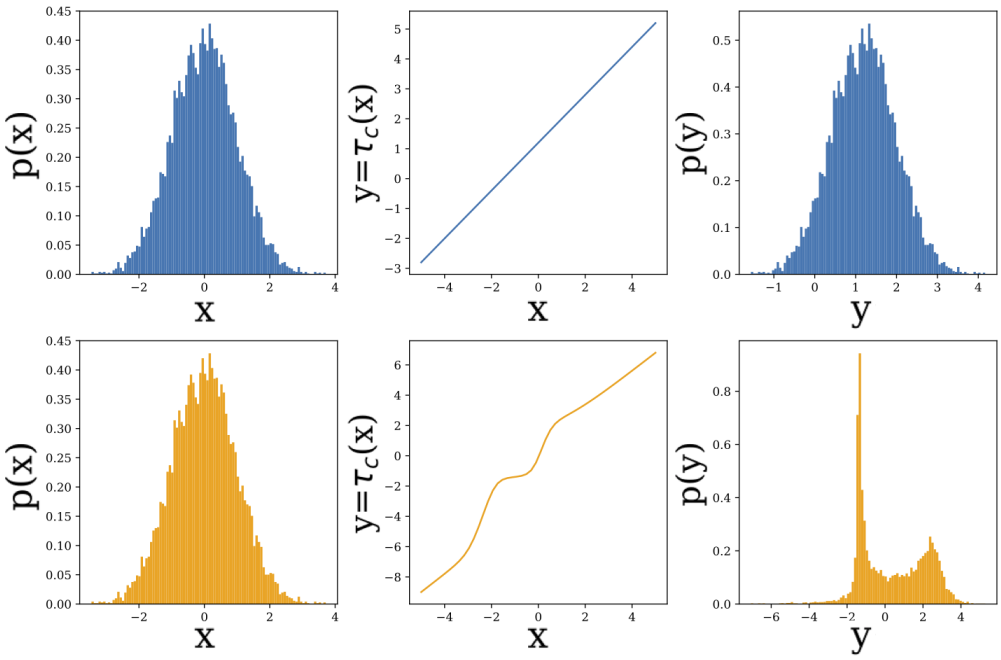


Figure 9.5: Illustration of the effects of traditional IAF (top), and our proposed NAF (bottom). Areas where the slope of the transformer τ_c is greater/less than 1, are compressed/expanded (respectively) in the output distribution. Inflection points in $\tau_c(x_t)$ (middle) can transform a unimodal $p(x_t)$ (left) into a multimodal $p(y_t)$ (right); NAF allows for such inflection points, whereas IAF does not.

efficiently via backpropagation (as per requirement 2).

Whereas affine transformers require information about multimodality in y_t to flow through $x_{1:t-1}$, our **neural autoregressive flows (NAFs)** are able to induce multimodality more naturally, via inflection points in τ_c, as shown in Figure 9.5. Intuitively, τ_c can be viewed as analogous to a cumulative distribution function (CDF), so that its derivative corresponds to a PDF, where its inflection points yield local maxima or minima.

9.3.1 Transformer architectures

In this work, we use two specific architectures for τ_c, which we refer to as **deep sigmoidal flows (DSF)** and **deep dense sigmoidal flows (DDSF)** (see Figure 9.4(b), 9.4(c) for an illustration). We find that small neural network transformers of 1 or 2 hidden layers with 8 or 16 sigmoid units perform well across our experiments, although there are other possibilities worth exploring (see Section 9.3.3). Sigmoids contain inflection points, and so can easily induce inflection points in τ_c, and thus multimodality in $p(y_t)$. We begin by describing the DSF transformation, which is already sufficiently expressive to form a universal approximator for probability distributions, as we prove in section 9.4.

The DSF transformation resembles an MLP with a single hidden layer of sigmoid units. Naive use of sigmoid activation functions would restrict the range of τ_c, however, and result in a model that assigns 0 density to sufficiently large or small y_t, which is problematic when y_t can take on arbitrary real values. We address this issue by applying the inverse sigmoid (or "logit") function at the output layer. To ensure that the output's preactivation is in the domain of the logit (that is, $(0,1)$), we combine the output of the sigmoid units via an attention-like [13] softmax-weighted sums:

$$y_t = \sigma^{-1}(\underbrace{w^T}_{1\times d} \cdot \sigma(\underbrace{a}_{d\times 1} \cdot \underbrace{x_t}_{1\times 1} + \underbrace{b}_{d\times 1})) \tag{9.8}$$

where $0 < w_{i,j} < 1$, $\sum_i w_{i,j} = 1$, $a_{s,t} > 0$, and d denotes the number of hidden units [10].

Since all of the sigmoid activations are bounded between 0 and 1, the final preactiva-

[10] Constraints on the variables are enforced via activation functions; w and a are outputs of a softmax, and softplus or exp, respectively.

tion (which is their convex combination) is as well. The complete DSF transformation can be seen as mapping the original random variable to a different space through an activation function, where doing affine/linear operations is non-linear with respect to the variable in the original space, and then mapping it back to the original space through the inverse activation.

When stacking multiple sigmoidal transformation, we realize it resembles an MLP with bottleneck as shown by the bottom left of Figure 9.4. A more general alternative is the **deep dense sigmoidal flow (DDSF)**, which takes the form of a fully connected MLP:

$$h^{(l+1)} = \sigma^{-1}\Big(\underbrace{w^{(l+1)}}_{d_{l+1}\times d_{l+1}} \cdot \sigma\big(\underbrace{a^{(l+1)}}_{d_{l+1}} \odot \underbrace{u^{(l+1)}}_{d_{l+1}\times d_l} \cdot \underbrace{h^{(l)}}_{d_l} + \underbrace{b^{(1+1)}}_{d_{l+1}}\big)\Big) \tag{9.9}$$

for $1 \leq l \leq L$ where $h_0 = x$ and $y = h_L$; $d_0 = d_L = 1$. We also require $\sum_j w_{ij} = 1, \sum_j u_{kj} = 1$ for all i,k, and all parameters except b to be positive.

We use either DSF (Equation 9.8) or DDSF (Equation 9.9) to define the transformer function τ in Equation 9.4. To compute the log-determinant of Jacobian in a numerically stable way, we need to apply log-sum-exp to the chain rule

$$\nabla_x y = \left[\nabla_{h^{(L-1)}} h^{(L)}\right]\left[\nabla_{h^{(L-2)}} h^{(L-1)}\right], \cdots, \left[\nabla_{h^{(0)}} h^{(1)}\right] \tag{9.10}$$

We elaborate more on the numerical stability in parameterization and computation of logarithmic operations in the supplementary materials.

9.3.2 Efficient parametrization of larger transformers

Multi-layer NAFs, such as DDSF, require c to output $\mathcal{O}(d^2)$ pseudo-parameters, where d is the number of hidden units in each layer of τ. As this is impractical for large d, we propose parametrizing τ with $\mathcal{O}(d^2)$ statistical parameters, but only $\mathcal{O}(d)$ pseudo-parameters which modulate the computation on a per-unit basis, using a technique such as conditional batch-normalization (CBN) [81]. Such an approach also makes it possible to use minibatch-style matrix-matrix products for the forward and backwards propagation

through the graph of τ_c. In particular, we use a technique similar to *conditional weight normalization (CWN)* [176] in our experiments with DDSF; see appendix for details.

9.3.3 Possibilities for alternative architectures

While the DSF and DDSF architectures performed well in our experiments, there are many alternatives to be explored. One possibility is using other (strictly) monotonic activation functions in τ_c, such as leaky ReLUs (LReLU) [312] or ELUs [59]. Leaky ReLUs in particular are bijections on $\mathbb{R}$ and so would not require the softmax-weighted summation and activation function inversion tricks discussed in the previous section.

Finally, we emphasize that in general, τ need not be expressed as a neural architecture; it only needs to satisfy the requirements of invertibility and differentiability given at the end of section 2.

9.4 NAFs are universal density approximators

In this section, we prove that NAFs (specifically DSF) can be used to approximate any probability distribution over real vectors arbitrarily well, given that τ_c has enough hidden units output by generic neural networks with autoregressive conditioning. Ours is the first such result we are aware of for finite normalizing flows.

Our result builds on the work of Huang et al. [148], who demonstrate the general universal representational capability of inverse autoregressive transformations parameterized by an autoregressive neural network (that transform uniform random variables into any random variables in reals). However, we note that their proposition is weaker than we require, as there are no constraints on the parameterization of the transformer τ, whereas we've constrained τ to have strictly positive weights and monotonic activation functions, to ensure it is invertible throughout training.

The idea of proving the universal approximation theorem for DSF (1) in the IAF direction (which transforms unstructured random variables into structured random variables) resembles the concept of the **inverse transform sampling**: we first draw a sample from a simple distribution, such as uniform distribution, and then pass the sample though

DSF. If DSF converges to any inverse conditional CDF, the resulting random variable then converges in distribution to any target random variable as long as the latter has positive continuous probability density everywhere in the reals. (2) For the MAF direction, DSF serves as a solution to the non-linear independent component analysis problem [152], which disentangles structured random variables into uniformly and independently distributed random variables. (3) Combining the two, we further show that DSF can transform any structured noise variable into a random variable with any desired distribution.

We define the following notation for the pre-logit of the DSF transformation (compare equation 9.8):

$$\mathscr{S}(x_t, \mathscr{C}(x_{1:t-1})) = \sum_{j=1}^{n} w_j(x_{1:t-1}) \cdot \sigma\left(\frac{x_t - b_j(x_{1:t-1})}{\tau_j(x_{1:t-1})}\right) \tag{9.11}$$

where $\mathscr{C} = (w_j, b_j, \tau_j)_{j=1}^n$ are functions of $x_{1:1-t}$ parameterized by neural networks. Let b_j be in (r_0, r_1); τ_j be bounded and positive; $\sum_{j=1}^n w_j = 1$ and $w_j > 0$. See Appendix for the proof.

Proposition 2. (DSF universally transforms uniform random variables into any desired random variables) *Let Y be a random vector in $\mathbb{R}^m$ and assume Y has a strictly positive and continuous probability density distribution. Let $X \sim \text{Unif}((0,1)^m)$. Then there exists a sequence of functions $(G_n)_{n\geq 1}$ parameterized by autoregressive neural networks in the following form*

$$G(\mathbf{x})_t = \sigma^{-1}\left(\mathscr{S}\left(x_t; \mathscr{C}_t(x_{1:t-1})\right)\right) \tag{9.12}$$

where $\mathscr{C}_t = (a_{tj}, b_{tj}, \tau_{tj})_{j=1}^n$ are functions of $x_{1:t-1}$, such that $Y_n \doteq G_n(X)$ converges in distribution to Y.

Proposition 3. (DSF universally transforms any random variables into uniformly distributed random variables) *Let X be a random vector in an open set $\mathscr{U} \subset \mathbb{R}^m$. Assume X has a positive and continuous probability density distribution. Let $Y \sim \text{Unif}((0,1)^m)$. Then there exists a sequence of functions $(H_n)_{n\geq 1}$ parameterized by autoregressive neural*

networks in the following form

$$H(\mathbf{x})_t = \mathscr{S}\left(x_t; \mathscr{C}_t(x_{1:t-1})\right) \tag{9.13}$$

where $\mathscr{C}_t = (a_{tj}, b_{tj}, \tau_{tj})_{j=1}^n$ *are functions of* $x_{1:t-1}$*, such that* $Y_n \doteq H_n(X)$ *converges in distribution to* Y*.*

Theorem 3. (DSF universally transforms any random variables into any desired random variables) *Let* X *be a random vector in an open set* $\mathscr{U}\ \mathbb{R}^m$*. Let* Y *be a random vector in* $\mathbb{R}^m$*. Assume both* X *and* Y *have a positive and continuous probability density distribution. Then there exists a sequence of functions* $(K_n)_{n\geq 1}$ *parameterized by autoregressive neural networks in the following form*

$$K(\mathbf{x})_t = \sigma^{-1}\left(\mathscr{S}\left(x_t; \mathscr{C}_t(x_{1:t-1})\right)\right) \tag{9.14}$$

where $\mathscr{C}_t = (a_{tj}, b_{tj}, \tau_{tj})_{j=1}^n$ *are functions of* $x_{1:t-1}$*, such that* $Y_n \doteq K_n(X)$ *converges in distribution to* Y*.*

9.5 Related work

Neural autoregressive flows are a generalization of the affine autoregressive flows introduced by Kingma et al. [167] as **inverse autoregressive flows (IAF)** and further developed by Chen et al. [51] and Papamakarios et al. [230] as **autoregressive flows (AF)** and **masked autoregressive flows (MAF)**, respectively; for details on their relationship to our work see Sections 2 and 3. While Dinh et al. [73] draw a particular connection between their **NICE** model and the **Neural Autoregressive Density Estimator (NADE)** [184], [167] were the first to highlight the general approach of using autoregressive models to construct normalizing flows. Chen et al. [51] and then Papamakarios et al. [230] subsequently noticed that this same approach could be used efficiently in reverse when the key operation is evaluating, as opposed to sampling from, the flow's learned output density. Our method increases the expressivity of these previous approaches by using a neural net to output pseudo-parameters of another network, thus falling into the

hypernetwork framework [25, 70, 126].

There has been a growing interest in normalizing flows (NFs) in the deep learning community, driven by successful applications and structural advantages they have over alternatives. Rippel and Adams [247], Rezende and Mohamed [244] and Dinh et al. [73] first introduced normalizing flows to the deep learning community as density models, variational posteriors and generative models, respectively. In contrast to traditional variational posteriors, NFs can represent a richer family of distributions without requiring approximations (beyond Monte Carlo estimation of the KL-divergence). The NF-based **RealNVP**-style generative models [73, 76] also have qualitative advantages over alternative approaches. Unlike **generative adversarial networks (GANs)** [113] and **varational autoencoders (VAEs)** [164, 246], computing likelihood is cheap. Unlike autoregressive generative models, such as **pixelCNNs** [294], sampling is also cheap. Unfortunately, in practice RealNVP-style models are not currently competitive with autoregressive models in terms of likelihood, perhaps due to the more restricted nature of the transformations they employ.

Several promising recent works expand the capabilities of NFs for generative modeling and density estimation, however. Perhaps the most exciting example is van den Oord et al. [295], who propose the **probability density distillation** technique to train an IAF [167] based on the autoregressive **WaveNet** [293] as a generative model using another pretrained WaveNet model to express the target density, thus overcoming the slow sequential sampling procedure required by the original WaveNet (and characteristic of autoregressive models in general), and reaching super-real-time speeds suitable for production. The previously mentioned MAF technique [230] further demonstrates the potential of NFs to improve on state-of-the-art autoregressive density estimation models; such highly performant MAF models could also be "distilled" for rapid sampling using the same procedure as in van den Oord et al. [295].

Other recent works also find novel applications of NFs, demonstrating their broad utility. Loaiza-Ganem et al. [200] use NFs to solve maximum entropy problems, rather than match a target distribution. Louizos and Welling [204] and Krueger et al. [176] apply NFs to express approximate posteriors over parameters of neural networks. Song

et al. [269] use NFs as a proposal distribution in a novel Metropolis-Hastings MCMC algorithm.

Finally, there are also several works which develop new techniques for constructing NFs that are orthogonal to ours [23, 82, 104, 284, 285].

Table 9.I: Using DSF to improve variational inference. We report the number of affine IAF with our implementation. We note that the log likelihood reported by Kingma et al. [167] is 78.88. The average and standard deviation are carried out with 5 trials of experiments with different random seeds.

Model	ELBO	$\log p(x)$
VAE	85.00 ± 0.03	81.66 ± 0.05
IAF-affine	82.25 ± 0.05	80.05 ± 0.04
IAF-DSF	81.92 ± 0.04	79.86 ± 0.01

Table 9.II: Test log-likelihood and error bars of 2 standard deviations on the 5 datasets (5 trials of experiments). Neural autoregressive flows (NAFs) produce state-of-the-art density estimation results on all 5 datasets. The numbers (5 or 10) in parantheses indicate the number of transformations which were stacked; for TAN [224], we include their best results, achieved using different architectures on different datasets. We also include validation results to give future researchers a fair way of comparing their methods with ours during development.

Model	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
MADE MoG	0.40 ± 0.01	8.47 ± 0.02	-15.15 ± 0.02	-12.27 ± 0.47	153.71 ± 0.28
MAF-affine (5)	0.14 ± 0.01	9.07 ± 0.02	-17.70 ± 0.02	-11.75 ± 0.44	155.69 ± 0.28
MAF-affine (10)	0.24 ± 0.01	10.08 ± 0.02	-17.73 ± 0.02	-12.24 ± 0.45	154.93 ± 0.28
MAF-affine MoG (5)	0.30 ± 0.01	9.59 ± 0.02	-17.39 ± 0.02	-11.68 ± 0.44	156.36 ± 0.28
TAN (various architectures)	0.48 ± 0.01	11.19 ± 0.02	-15.12 ± 0.02	-11.01 ± 0.48	157.03 ± 0.07
MAF-DDSF (5)	$\mathbf{0.62 \pm 0.01}$	11.91 ± 0.13	$\mathbf{-15.09 \pm 0.40}$	$\mathbf{-8.86 \pm 0.15}$	$\mathbf{157.73 \pm 0.04}$
MAF-DDSF (10)	0.60 ± 0.02	$\mathbf{11.96 \pm 0.33}$	-15.32 ± 0.23	-9.01 ± 0.01	157.43 ± 0.30
MAF-DDSF (5) valid	0.63 ± 0.01	11.91 ± 0.13	15.10 ± 0.42	-8.38 ± 0.13	172.89 ± 0.04
MAF-DDSF (10) valid	0.60 ± 0.02	11.95 ± 0.33	15.34 ± 0.24	-8.50 ± 0.03	172.58 ± 0.32

9.6 Experiments

Our experiments evaluate NAFs on the classic applications of variational inference and density estimation, where we outperform IAF and MAF baselines. We first demonstrate

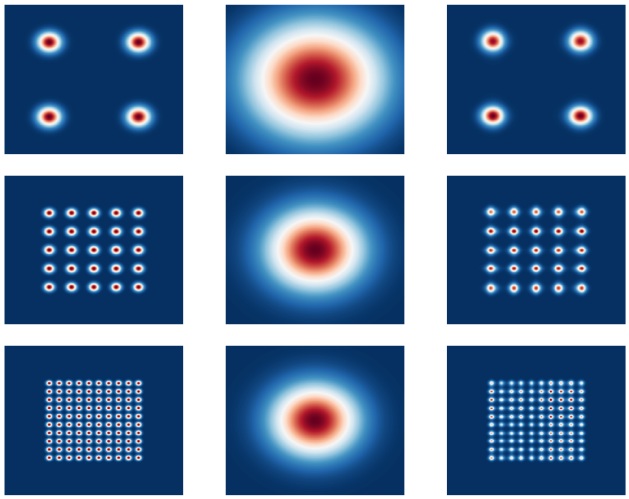


Figure 9.6: Fitting grid of Gaussian distributions using maximum likelihood. Left: true distribution. Center: affine autoregressive flow (AAF). Right: neural autoregressive flow (NAF)

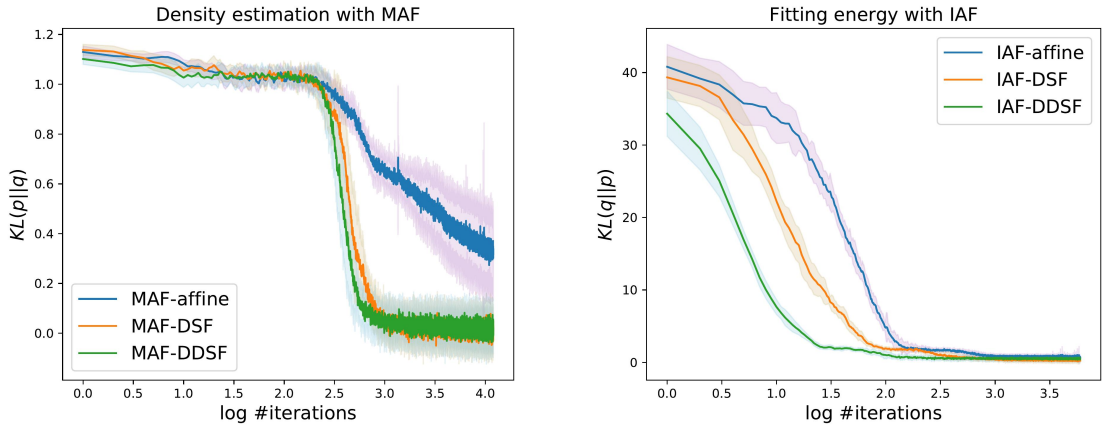


Figure 9.7: Learning curve of MAF-style and IAF-style training. q denotes our trained model, and p denotes the target.

the qualitative advantage NAFs have over AAFs in energy function fitting and density estimation (Section 9.6.1). We then demonstrate the capability of NAFs to capture a multimodal Bayesian posterior in a limited data setting (Section 9.6.2). For larger-scale experiments, we show that using NAF instead of IAF to approximate the posterior distribution of latent variables in a variational autoencoder [164, 246] yields better likelihood results on binarized MNIST [184] (Section 9.6.3). Finally, we report our

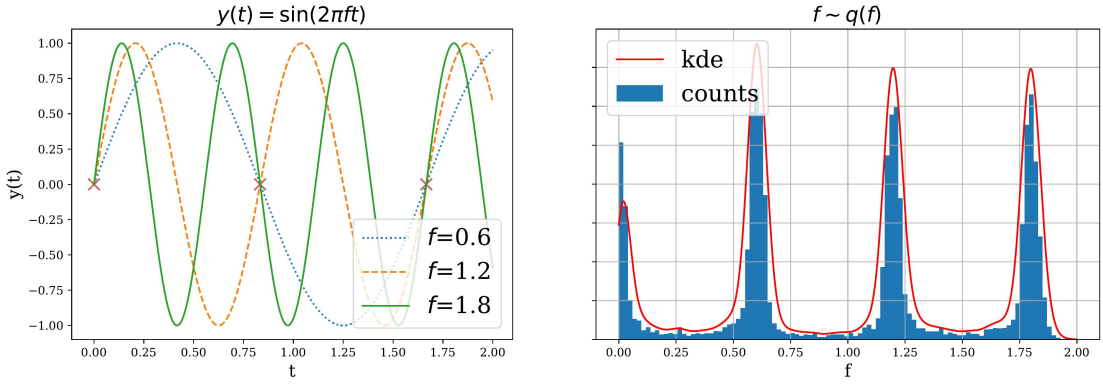


Figure 9.8: The DSF model effectively captures the true posterior distribution over the frequency of a sine wave. Left: The three observations (marked with red x's) are compatible with sine waves of frequency $f \in 0.0, 0.6, 1.2, 1.8$. Right: a histogram of samples from the DSF approximate posterior ("counts") and a Kernel Density Estimate of the distribution it represents (KDE).

experimental results on density estimation of a suite of UCI datasets (Section 9.6.4).

9.6.1 Toy energy fitting and density estimation

9.6.1.0.1 Expressiveness. First, we demonstrate that, in the case of marginally independent distributions, affine transformation can fail to fit the true distribution. We consider a mixture of Gaussian density estimation task. We define the modes of the Gaussians to be laid out on a 2D meshgrid within the range [-5,5], and consider 2, 5 and 10 modes on each dimension. While the affine flow only produces a single mode, the neural flow matches the target distribution quite well even up to a 10x10 grid with 100 modes (see Figure 9.6).

9.6.1.0.2 Convergence. We then repeat the experiment that produces Figure 9.1 and 9.2 16 times, smooth out the learning curve and present average convergence result of each model with its corresponding standard deviation. For affine flow, we stack 6 layers of transformation with reversed ordering. For DSF and DDSF we used one transformation. We set $d = 16$ for both, $L = 2$ for DDSF.

9.6.2 Sine wave experiment

Here we demonstrate the ability of DSF to capture multimodal posterior distributions. To do so, we create a toy experiment where the goal is to infer the posterior over the frequency of a sine wave, given only 3 datapoints. We fix the form of the function as $y(t) = \sin(2\pi f \cdot t)$ and specify a Uniform prior over the frequency: $p(f) = U([0,2])$. The task is to infer the posterior distribution $p(f|T,Y)$ given the dataset $(T,Y) = ((0,5/6,10/6),(0,0,0))$, as represented by the red crosses of Figure 9.8 (left). We assume the data likelihood given the frequency parameter to be $p(y_i|t_i,f) = \mathcal{N}(y_i; y_f(t_i), 0.125)$, where the variance $\sigma^2 = 0.125$ represents the inherent uncertainty of the data. Figure 9.8 (right) shows that DSF learns a good posterior in this task.

9.6.3 Amortized approximate posterior

We evaluate NAF's ability to improve variational inference, in the context of the binarized MNIST [184] benchmark using the well-known variational autoencoder [164, 246] (Table 9.I). Here again the DSF architecture outperforms both standard IAF and the traditional independent Gaussian posterior by a statistically significant margin.

9.6.4 Density estimation with masked autoregressive flows

We replicate the density estimation experiments of Papamakarios et al. [230], which compare MADE [105] and RealNVP [76] to their proposed MAF model (using either 5 or 10 layers of MAF) on BSDS300 [211] as well as 4 UCI datasets [195] processed as in Uria et al. [292]. Simply replacing the affine transformer with our DDSF architecture in their best performing architecture for each task (keeping all other settings fixed) results in substantial performance gains, and also outperforms the more recent Transformation Autoregressive Networks (TAN) Oliva et al. [224], setting a new state-of-the-art for these tasks. Results are presented in Table 9.II.

9.7 Conclusion

In this work we introduce the neural autoregressive flow (NAF), a flexible method of tractably approximating rich families of distributions. In particular, our experiments show that NAF is able to model multimodal distributions and outperform related methods such as inverse autoregressive flow in density estimation and variational inference. Our work emphasizes the difficulty and importance of capturing multimodality, as previous methods fail even on simple toy tasks, whereas our method yields significant improvements in performance.

Acknowledgements

We would like to thank Tegan Maharaj, Ahmed Touati, Shawn Tan and Giancarlo Kerg for helpful comments and advice. We also thank George Papamakarios for providing details on density estimation task's setup.

CHAPTER 10

CONCLUSION

The articles presented in this thesis make progress on understanding and improving the generalization abilities of Deep Learning, and on probabilistic and Bayesian methods of Deep Learning. Generalization is of fundamental importance to AI Alignment as argued in Section 1.4.1. Out-of-distribution generalization in particular (the subject of the work presented in Chapter 4) remains a critical weakness of AI methods including deep learning. Failures of out-of-distribution generalization are a major source of concerns about AI x-risk. Bayesian deep learning continues to be a potentially promising complementary approach both for improving generalization and addressing or mitigating generalization failures (via active learning and error detection, respectively).

This work has focused not only on the technical contributions of these works, but also on elaborating the motivations discussed in brief in the above paragraph. Connecting narrow technical research problems to the goal of increasing AI x-safety led us through a deep and involved chain of reasoning, often invoking and bottoming out in somewhat tenuous and/or philosophical positions. A more complete and rigorous defense of these positions goes far beyond the scope of this thesis, but I hope I've managed to communicate most of the key intuitions and provided some compelling reasons for them.

Section 10.1 of this conclusion we will briefly revisit our motivation of increasing AI x-safety, and reevaluate our contributions from this perspective. Then in Section 10.2 we will look forward at how the underlying goals of these works might best be pursued given the current research landscape. In my current estimation, these goals remain relevant and important for x-safety, although the specific means of pursuing them that I have employed seem less promising in light of recent developments. The neglectedness of these goals has also decreased, making work on them less appealing from the perspective of counterfactual impact. With all of this in mind, in Section 10.3, I will also offer some thoughts on what areas of research seem particularly valuable at present.

10.1 Revisiting the motivation for working on AI x-safety, and how my work fits in

Achieving a low level of AI x-risk without forgoing the development of AGI seems likely to require significant advances in AI alignment (as well as other areas). Without the ability to reliably direct AI systems towards behaviors their designers endorse, the continued development and deployment of increasingly advanced and ultimately super-human AI systems seems bound to disempower humans in our efforts to influence the future, achieve our goals, and realize our values. Furthermore, it seems irresponsible to avoid grappling proactively with the AI alignment challenges we face and can expect to face, since we cannot predict the timeline on which advances in AI will occur, or be confident in our ability to address social problems or mitigate existential risks in a timely manner as they arise.

While the control over an AI system wielded by its developers is considerable, it by no means guarantees that the system will behave as the developers intend. The inscrutable nature of modern AI systems means that in practice we do not know how well the system's behavior matches the intended behavior, and instead are left to rely on limited evaluations and proxy metrics. Even worse, machine learning systems are trained to optimize such proxy metrics, despite well known limitations of such methodology.

The line between misuse of AI and accidents caused by AI systems is inherently blurry; more responsible developers will take greater efforts to ensure that their systems behave as intended. The risks imposed by AI systems on society, including existential risk, are an externality whose cost is not currently well accounted for. While this remains the case, we can expect that many developers will take reckless risks, choosing to deploy AI systems that are not sufficiently aligned, as well as calculated risks, deploying AI systems that impose a socially sub optimal level of x risk. AI alignment does not directly address the issue of calculated risks, but it could reduce reckless risk taking and help us leverage capabilities to address the social dilemma posed by existential risks.

10.1.1 Alignment and generalization

Alignment of advanced systems seems difficult to achieve without their having an understanding of various human concepts. The conceptual and empirical promise of deep learning systems as a means of learning human concepts makes understanding their generalization and learning behavior important for achieving this end. This is especially true in light of various phenomena, such as adversarial examples and memorization, that bely their impressive ability to generalize in distribution. Indeed, what is needed, and and would be by indicative of genuine understanding of human concepts, is out-of-distribution generalization. The works presented in this thesis provide insight into to these topics. Our extensive experiments in "A Closer Look at Memorization" (Chapter 2) demonstrated a number of novel learning phenomena and introduced the hypothesis that Deep Learning learns simple patterns first. Meanwhile, "Out-of-Distribution Generalization via Risk Extrapolation (REx)" (Chapter 4) provided a novel geometric interpretation of invariant prediction, demonstrated a limitation of invariant prediction in addressing covariate shift (i.e. change in $P(X)$), and underlined the theoretical importance of OOD generalization techniques (as opposed to, e.g. simply scaling up the size of the dataset and/or model).

10.1.2 Managing uncertainty

Another property that seems crucial for a AI alignment is appropriate management of uncertainty. Arguably the most important tool for managing uncertainty is probability theory. Any AI system operating in a complex real-world environment will have to make decisions about how to manage trade-offs between different risks, based on their probability and magnitude. The ability to approximate probability distributions arbitrarily well, exhibited by Neural Autoregressive Flows (Chapter 8), is a key desiderata.

Bayesian probability theory adds to the probabilistic treatment of uncertainty by allowing one to distinguish between uncertainty due to ignorance and uncertainty due to inherent randomness. This distinction can help improve the efficiency of learning and may also provide useful for recognizing when to behave cautiously. Bayesian Hypernetworks (Chapter 6) are a method of approximating the Bayesian posterior of a

DNN with another DNN, which provides advantages over simpler approximate posteriors, such as the potential for multi-modality.

10.2 What are the best ways to achieve the goals of good/cautious generalization?

Putting it all together, the ultimate goal or vision of these works is an AI system that understands human concepts necessary for aligned behaviour (good generalization), or is able to recognise its failure to do so, especially when this lack of understanding leads to large uncertainty or risk (cautious generalization). An important question is whether the approaches I have pursued are the most promising for achieving that goal. Here I will provide some reasons for suspecting that they are not, and discuss some alternatives.

While my research has primarily focused on deep learning, I suspect that some forms of generalization will require other methods, e.g. the ability to learn to extrapolate using symbolic expressions. This is not something deep learning is fundamentally incapable of, but it is beyond the ability of currently popular neural network architectures. I believe a hybrid approach, which could look as simple as using a neural network to to generate and manipulate symbolic expressions, is likely to be necessary in order to achieve the kinds of out-of-distribution generalization that we seek, such as systematic generalization.

On the other hand, the causal perspective on out-of-distribution generalization provided by invariant prediction may help us diagnose existing generalisation failures that are unlikely to be solved via scaling or even better choices of model or learning algorithm. This is because these failures may ultimately lie in the data collection process. In REx (Chapter 4), we demonstrate that data diversity is not a sufficient condition for out-of-distribution generalisation. Rather, the details of the data distribution remain critical. Further study along these lines may help elucidate fundamental challenges for and limitations of current learning paradigms.

For the time being, however, increasing data diversity and scaling to larger deep learning models with higher performance in-distribution often seems to bring the largest benefit for out-of-distribution performance [137].

Stepping back from the goal of generalization, good out-of-distribution behavior could

also be achieved via error detection, out-of-distribution detection, or other approaches to conservative behaviour with respect to known or suspected risks. Conceptually, this seems more tractable then generalizing appropriately (see Section 1.4.1). This was also part of the motivation for my interest in Bayesian methods. However, Bayesian Deep Learning has so far failed to live up to its promise empirically. One reason for this is their (seeming) inability to explore different modes of the posterior [93]. This makes ensembles a more appealing alternative in practice for the time being. With respect to OOD detection in particular, scaling and transfer appear extremely promising, when large training sets are available [94] Overall, it seems more promising at the moment to pursue alternative approaches to achieving the functionalities provided by Bayesian methods.

In the relatively narrow field of normalising flows, several fundamentally new approaches to constructing normalising flow models have been proposed that seem more promising at present than autoregressive flows for most applications [50, 120].

In summary, I believe that the most durable legacy of these four works will be insights they provided, especially in their particular historical context, rather than the specific methods we proposed.

10.3 Some research directions for AI x-safety that seem particularly valuable at present

The field of AI moves fast, and topics that were almost entirely neglected a year ago might garner hundreds of submissions at the next conference. This is true of many topics relevant for AI alignment, including those included in this thesis. I will now highlight a few topics that seem more timely to study.

Perhaps the most significant recent development in AI is the rapid scaling of unsupervised Deep Learning using Transformers, as exemplified by GPT-3 [39], which cost roughly 10 million USD to train, and achieved state-of-the-art performance on a number of tasks without any fine-tuning. This suggests that such large models, dubbed "foundation models" [30] are likely to form a major part of many AI systems for the foreseeable future – that is, assuming developers have access to them; the cost of training

such models may be prohibitive for all but the largest tech companies. In any case, these models have much more impressive capabilities than smaller deep learning models, and further scaling is expected to yield further improvements in capabilities [158]. I will not speculate (much) on the implications of this new 'paradigm', but understanding and aligning foundation models is a critical task for increasing AI x-safety, and seems likely to create new challenges and opportunities. Fortunately, from the point of view of x-safety, it seems to be possible to control their behavior quite a bit via prompt-engineering [242] or fine-tuning, e.g. based on reward modeling [272] or instructions [303].

Foundation models also make the *need* for alignment apparent, as they appear to have latent capabilities which require alignment techniques to access [48]. The emergence of alignment as a practical problem has been and will be accompanied by the use of alignment techniques such as reward modelling. However, the demands that practical applications place on these techniques are less than what is required for x-safety (see Section 1.3.4). This creates a need for research that anticipates their limitations and failure modes. The significance of the distinction between aligned behavior and the *apparently* aligned behavior is exemplified by the notion of a "treacherous turn" – a 'deliberate' act of deception on the part of an AI system [34]. However, this is merely a particularly extreme and distinctive form of a more general problem that now urgently needs addressing.

In this work, I have focused on what is is often called the outer alignment problem. This is to distinguish it from the less well known, more speculative, inner alignment problem, which is fart to be perhaps more insidious and pervasive. As the outer alignment problem becomes more widely recognised and studied, inner alignment, which is still not discussed much outside of the alignment research community, becomes relatively more neglected and therefore useful to study. The development of foundation models marks a trend away from reinforcement learning in favour of unsupervised learning; this also increases the importance of inner alignment, since alignment failures are particularly concerning when the misaligned AI is an agent that seeks to influence the world in order to achieve it's goals (as in RL). Inner alignment concerns include the possibility that such goal-directed, agent-like behaviour could emerge from AI systems not trained to seek such influence, e.g. those trained with (un)supervised learning or other myopic learning

approaches. Our work, Krueger et al. [179], addresses a similar concern. Understanding the potential for other forms of emergent agency seems critical and continues to be neglected. For instance, the interactions of multiple humans and multiple AI systems could create emergent dynamics that threaten the agency of humans, as discussed in Section 1.2.4.

Finally, as awareness of negative social impacts of AI has increased, along with political will to address them, the value of technical work that could lead to enforceable standards for the behavior of AI systems is increasing as well. Possible examples include myopia [179] or truthfulness [88]. In a similar spirit, Brundage et al. [41] argue for the need to go beyond ethical principles to implementing mechanisms for trustworthy AI.

10.4 Final summary

In this thesis, I've discussed the problem of existential risk from AI ("AI x-risk") and how it might be mitigated. In particular, I've focused on how AI Alignment work grounded in Machine Learning may help contribute to reducing AI x-risk, although my position is that it is unlikely to prove sufficient to reduce this risk to an acceptable level.

Still, there is much work to do in AI Alignment, and the topic remains somewhat neglected for the time being. However, the specific areas of research the articles in this thesis covered no longer seem to be neglected. The adoption of particular research topics relevant to AI x-safety is a good thing, likewise the increasing recognition that AI contributes to x-risk.

As this trend (hopefully) continues, a rational response for a researcher aiming to reduce x-risk may be to focus on increasingly speculative or arcane concerns that have yet to capture the attention of the broader research community. Other possible responses include engaging in more applied work aiming to ensure that specific AI systems or developers are not increasing x-risk due to oversight, or engaging in policy work to increase the adoption of best-practices for x-safety. Finally, there remain important outstanding fundamental questions about how we can increase our justified confidence in advanced AI systems' alignment.

All things considered, I think aiming to increase AI x-safety will continue being a useful way to orient and prioritize research, and AI Alignment will continue to be a fruitful approach. I expect the popularity and acceptance of both of these topics to continue to grow, both due to technical necessity and due to increasing obligations on AI researchers and developers to consider the social impact of their work.

BIBLIOGRAPHY

[1] Scott Aaronson. Why I Am Not An Integrated Information Theorist (or, The Unconscious Expander), 2014. https://www.scottaaronson.com/blog/?p=1799.

[2] Isabela Albuquerque, Nikhil Naik, Junnan Li, Nitish Keskar, and Richard Socher. Improving out-of-distribution generalization via multi-task self-supervised pretraining, 2020.

[3] Scott Alexander. Meditations on Moloch. Slate Star Codex, 2014. https://slatestarcodex.com/2014/07/30/meditations-on-moloch/.

[4] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete Problems in AI Safety. *CoRR*, abs/1606.06565, 2016. URL http://arxiv.org/abs/1606.06565.

[5] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete Problems in AI Safety, 2016.

[6] Guozhong An. The effects of adding noise during backpropagation training on a generalization performance. *Neural computation*, 8(3):643–674, 1996.

[7] arbital.com. Optimization daemons, 2021. https://arbital.com/p/daemons/.

[8] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[9] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A Closer Look at Memorization in Deep Networks, 2017.

[10] Amanda Askell. Pareto Principles in Infinite Ethics. Master's thesis, New York University, 2018.

[11] Philip Bachman, R. Devon Hjelm, and William Buchwalter. Learning Representations by Maximizing Mutual Information Across Views, 2019.

[12] J. Andrew Bagnell. Robust Supervised Learning. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*, AAAI'05, pages 714–719. AAAI Press, 2005. ISBN 157735236x.

[13] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

[14] Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron C. Courville. Systematic Generalization: What Is Required and Can It Be Learned? *CoRR*, abs/1811.12889, 2018. URL `http://arxiv.org/abs/1811.12889`.

[15] Peter L. Bartlett, Olivier Bousquet, Shahar Mendelson, et al. Local rademacher complexities. *The Annals of Statistics*, 33(4):1497–1537, 2005.

[16] Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in Terra Incognita. *Lecture Notes in Computer Science*, pages 472–489, 2018. ISSN 1611-3349.

[17] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

[18] Shai Ben-David, Tyler Lu, Teresa Luu, and Dávid Pál. Impossibility theorems for domain adaptation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 129–136, 2010.

[19] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*, volume 28. Princeton University Press, 2009.

[20] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[21] Yoshua Bengio et al. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[22] Rob Bensinger. New paper: "Formalizing convergent instrumental goals". MIRI Machine Intelligence Research Institute, 2015. `https://intelligence.org/2015/11/26/new-paper-formalizing-convergent-instrumental-goals/`.

[23] Rianne van den Berg, Leonard Hasenclever, Jakub M. Tomczak, and Max Welling. Sylvester Normalizing Flows for Variational Inference. *arXiv preprint arXiv:1803.05649*, 2018.

[24] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Ponde de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with Large Scale Deep Reinforcement Learning, 2019.

[25] Luca Bertinetto, João F. Henriques, Jack Valmadre, Philip Torr, and Andrea Vedaldi. Learning feed-forward one-shot learners. In *Advances in neural information processing systems*, pages 523–531, 2016.

[26] Alexander Bird and Emma Tobin. Natural Kinds. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Spring 2018 edition, 2018.

[27] Chris M. Bishop. Training with noise is equivalent to Tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

[28] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1613–1622, 2015.

[29] P. Bojanowski and A. Joulin. Unsupervised Learning by Predicting Noise. *ArXiv e-prints*, April 2017.

[30] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, et al. On the Opportunities and Risks of Foundation Models. *CoRR*, abs/2108.07258, 2021. URL `https://arxiv.org/abs/2108.07258`.

[31] Nick Bostrom. How Long Before Superintelligence? *International Journal of Futures Studies*, 2, 1998.

[32] Nick Bostrom. Existential risks: Analyzing human extinction scenarios and related hazards. *Journal of Evolution and technology*, 9(1), 2002.

[33] Nick Bostrom. The superintelligent will: Motivation and instrumental rationality in advanced artificial agents. *Minds and Machines*, 22(2):71–85, 2012.

[34] Nick Bostrom. Superintelligence: Paths, Dangers, Strategies, 2014.

[35] Nick Bostrom, Thomas Douglas, and Anders Sandberg. The Unilateralist's Curse

and the Case for a Principle of Conformity. *Social epistemology*, 30(4):350–371, 2016.

[36] Nick Bostrom et al. Infinite ethics. *Analysis and Metaphysics*, 10:9–59, 2011.

[37] Léon Bottou. Online learning and stochastic approximations. *On-line learning in neural networks*, 17(9):142, 1998.

[38] Tom B. Brown, Nicholas Carlini, Chiyuan Zhang, Catherine Olsson, Paul Christiano, and Ian Goodfellow. Unrestricted adversarial examples, 2018.

[39] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners, 2020.

[40] Wessel Bruinsma, Andrew Y. K. Foong, and Richard E. Turner. What Keeps A Bayesian Awake At Night? Part 2: Night Time. Cambridge MLG Blog, 2021. `bruinsma2021what`.

[41] Miles Brundage, Shahar Avin, Jasmine Wang, Haydn Belfield, Gretchen Krueger, Gillian Hadfield, Heidy Khlaaf, Jingying Yang, Helen Toner, Ruth Fong, Tegan Maharaj, Pang Wei Koh, Sara Hooker, Jade Leung, Andrew Trask, Emma Bluemke, Jonathan Lebensold, Cullen O'Keefe, Mark Koren, Théo Ryffel, JB Rubinovitz, Tamay Besiroglu, Federica Carugati, Jack Clark, Peter Eckersley, Sarah de Haas, Maritza Johnson, Ben Laurie, Alex Ingerman, Igor Krawczuk, Amanda Askell, Rosario Cammarota, Andrew Lohn, David Krueger, Charlotte Stix, Peter Henderson, Logan Graham, Carina Prunkl, Bianca Martin, Elizabeth Seger, Noa Zilberman, Seán Ó hÉigeartaigh, Frens Kroeger, Girish Sastry, Rebecca Kagan, Adrian Weller, Brian Tse, Elizabeth Barnes, Allan Dafoe, Paul Scharre, Ariel

Herbert-Voss, Martijn Rasser, Shagun Sodhani, Carrick Flynn, Thomas Krendl Gilbert, Lisa Dyer, Saif Khan, Yoshua Bengio, and Markus Anderljung. Toward Trustworthy AI Development: Mechanisms for Supporting Verifiable Claims, 2020.

[42] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance Weighted Autoencoders, 2016.

[43] Peter Bühlmann. Invariance, Causality and Robustness, 2018.

[44] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14, 2017.

[45] Joseph Carlsmith. Is Power-Seeking AI an Existential Risk?, 2021. `https://d8737ecf-376e-4788-8d12-a097599c13f6.filesusr.com/ugd/5f37c1_5333aa0b7ff7461abc208b25bfc7df87.pdf`.

[46] Fabio M. Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2229–2238, 2019.

[47] Pratik Chaudhari, Anna Choromanska, Stefano Soatto, and Yann LeCun. Entropy-SGD: Biasing Gradient Descent Into Wide Valleys. *arXiv preprint arXiv:1611.01838*, 2016.

[48] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss,

Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating Large Language Models Trained on Code, 2021.

[49] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed Chi. Top-K Off-Policy Correction for a REINFORCE Recommender System, 2020.

[50] Ricky T. Q. Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual Flows for Invertible Generative Modeling, 2020.

[51] Xi Chen, Diederik P. Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. Variational lossy autoencoder. In *International Conference on Learning Representations*, 2017.

[52] Xi Chen, Nikhil Mishra, Mostafa Rohaninejad, and Pieter Abbeel. PixelSNAIL: An Improved Autoregressive Generative Model, 2017.

[53] François Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

[54] Brian Christian. *The Alignment Problem: How Can Artificial Intelligence Learn Human Values?* Atlantic Books, 2021.

[55] Paul Christiano. Humans consulting HCH. AI Alignment, 2016. `https://ai-alignment.com/humans-consulting-hch-f893f6051455`.

[56] Paul Christiano. Strong HCH. AI Alignment, 2016. `https://ai-alignment.com/strong-hch-bedb0dc08d4e`.

[57] Paul Christiano. Corrigibility. AI Alignment, 2017. `https://ai-alignment.com/corrigibility-3039e668638`.

[58] Jack Clark and Dario Amodei. Faulty Reward Functions in the Wild. OpenAI Codex, 2016. https://openai.com/blog/faulty-reward-functions/.

[59] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). *International Conference on Learning Representations*, 2016.

[60] Ajeya Cotra. Why AI alignment could be hard with modern deep learning. Cold Takes, 2021. https://www.cold-takes.com/why-ai-alignment-could-be-hard-with-modern-deep-learning/.

[61] Kate Crawford. Artificial Intelligence's White Guy Problem. The New York Times, 2016. https://www.nytimes.com/2016/06/26/opinion/sunday/artificial-intelligences-white-guy-problem.html.

[62] Andrew Critch. What Multipolar Failure Looks Like, and Robust Agent-Agnostic Processes (RAAPs). LESSWRONG, 2021. https://www.lesswrong.com/posts/LpM3EAakwYdS6aRKf/what-multipolar-failure-looks-like-and-robust-agent-agnostic.

[63] Andrew Critch and David Krueger. AI Research Considerations for Human Existential Safety (ARCHES), 2020.

[64] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. AutoAugment: Learning Augmentation Policies from Data, 2018.

[65] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.

[66] Alexander D'Amour, Katherine Heller, Dan Moldovan, Ben Adlam, Babak Alipanahi, Alex Beutel, Christina Chen, Jonathan Deaton, Jacob Eisenstein, Matthew D. Hoffman, Farhad Hormozdiari, Neil Houlsby, Shaobo Hou, Ghassen Jerfel, Alan Karthikesalingam, Mario Lucic, Yian Ma, Cory McLean, Diana Mincu,

Akinori Mitani, Andrea Montanari, Zachary Nado, Vivek Natarajan, Christopher Nielson, Thomas F. Osborne, Rajiv Raman, Kim Ramasamy, Rory Sayres, Jessica Schrouff, Martin Seneviratne, Shannon Sequeira, Harini Suresh, Victor Veitch, Max Vladymyrov, Xuezhi Wang, Kellie Webster, Steve Yadlowsky, Taedong Yun, Xiaohua Zhai, and D. Sculley. Underspecification Presents Challenges for Credibility in Modern Machine Learning, 2020.

[67] DARPA. AlphaDogfight Trials Foreshadow Future of Human-Machine Symbiosis. DARPA.MIL, 2020. https://www.darpa.mil/news-events/2020-08-26.

[68] Shai Ben David, Tyler Lu, Teresa Luu, and Dávid Pál. Impossibility theorems for domain adaptation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 129–136. JMLR Workshop and Conference Proceedings, 2010.

[69] Peter de Blanc. Ontological Crises in Artificial Agents' Value Systems, 2011.

[70] Bert De Brabandere, Xu Jia, Tinne Tuytelaars, and Luc Van Gool. Dynamic Filter Networks. *CoRR*, abs/1605.09673, 2016. URL http://arxiv.org/abs/1605.09673.

[71] Abram Demski and Scott Garrabrant. Embedded Agency, 2020.

[72] Guillaume Desjardins, Karen Simonyan, Razvan Pascanu, et al. Natural neural networks. In *Advances in Neural Information Processing Systems*, pages 2071–2079, 2015.

[73] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[74] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear Independent Components Estimation, 2015.

[75] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density Estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

[76] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *International Conference on Learning Representations*, 2017.

[77] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP, 2017.

[78] Roel Dobbe, Thomas Krendl Gilbert, and Yonatan Mintz. Hard Choices in Artificial Intelligence: Addressing Normative Uncertainty through Sociotechnical Commitments. *CoRR*, abs/1911.09005, 2019. URL `http://arxiv.org/abs/1911.09005`.

[79] Michele Donini, Luca Oneto, Shai Ben-David, John Shawe-Taylor, and Massimiliano Pontil. Empirical Risk Minimization under Fairness Constraints, 2018.

[80] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, 2021.

[81] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *International Conference on Learning Representations*, volume 2, 2017.

[82] David Duvenaud, Dougal Maclaurin, and Ryan Adams. Early stopping as nonparametric variational inference. In *Artificial Intelligence and Statistics*, 2016.

[83] Peter Eckersley. Impossibility and uncertainty theorems in AI value alignment (or why your AGI should not have a utility function). *CoRR*, abs/1901.00064, 2019. URL `http://arxiv.org/abs/1901.00064`.

[84] Adrien Ecoffet and Joel Lehman. Reinforcement learning under moral uncertainty. In *International Conference on Machine Learning*, pages 2926–2936. PMLR, 2021.

[85] Effective Altruism Forum. ITN framework. Effective Altruism Forum, 2021. https://forum.effectivealtruism.org/tag/itn-framework-1.

[86] Jeffrey L. Elman. Finding Structure in Time. *Cognitive Science*, 14(2):179–211, 1990.

[87] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. *arXiv preprint arXiv:1712.02779*, 2017.

[88] Owain Evans, Owen Cotton-Barratt, Lukas Finnveden, Adam Bales, Avital Balwit, Peter Wills, Luca Righetti, and William Saunders. Truthful AI: Developing and governing AI that does not lie, 2021.

[89] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust Physical-World Attacks on Deep Learning Models, 2018.

[90] Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation, 2020.

[91] Evelyn Fix and Joseph L. Hodges Jr. Discriminatory analysis-nonparametric discrimination: consistency properties. Technical report, DTIC Document, 1951.

[92] Nic Ford, Justin Gilmer, Nicholas Carlini, and Ekin Dogus Cubuk. Adversarial Examples Are a Natural Consequence of Test Error in Noise. *CoRR*, abs/1901.10513, 2019. URL http://arxiv.org/abs/1901.10513.

[93] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep Ensembles: A Loss Landscape Perspective, 2020.

[94] Stanislav Fort, Jie Ren, and Balaji Lakshminarayanan. Exploring the Limits of Out-of-Distribution Detection. *CoRR*, abs/2106.03004, 2021. URL `fort2021 exploring`.

[95] Future of Life. An Open Letter: Research Priorities for Robust and Beneficial Artificial Intelligence, 2021. `https://futureoflife.org/2015/10/27/ai-open-letter/`.

[96] Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

[97] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

[98] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning, 2016.

[99] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian Active Learning with Image Data. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, pages 1183–1192, 2017. URL `http://proceedings.mlr.press/v70/gal17a.html`.

[100] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[101] Scott Garrabrant. Goodhart Taxonomy. LESSWRONG, 2017. `https://www.lesswrong.com/posts/EbFABnst8LsidYs5Y/goodhart-taxonomy`.

[102] Scott Garrabrant, Tsvi Benson-Tilsen, Andrew Critch, Nate Soares, and Jessica Taylor. Logical Induction, 2020.

[103] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. *arXiv preprint arXiv:1811.12231*, 2018.

[104] Mevlana C. Gemici, Danilo Jimenez Rezende, and Shakir Mohamed. Normalizing flows on riemannian manifolds. *arXiv preprint arXiv:1611.02304*, 2016.

[105] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: masked autoencoder for distribution estimation. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 881–889, 2015.

[106] Allan Gibbard. Manipulation of voting schemes: a general result. *Econometrica: journal of the Econometric Society*, pages 587–601, 1973.

[107] Justin Gilmer, Ryan P. Adams, Ian J. Goodfellow, David Andersen, and George E. Dahl. Motivating the Rules of the Game for Adversarial Example Research, 2018.

[108] Corrado Gini. Variabilita e Mutabilita. *Journal of the Royal Statistical Society*, 76 (3), 1913.

[109] Ian Goodfellow. The case for dynamic defenses against adversarial examples, 2019.

[110] Ian Goodfellow. A research agenda: Dynamic models to defend against correlated attacks, 2019.

[111] Ian J. Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. *CoRR*, abs/1701.00160, 2017. URL `http://arxiv.org/abs/1701.00160`.

[112] Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[113] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, 2014.

[114] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2672–2680, 2014. URL http://papers.nips.cc/paper/5423-generative-adversarial-nets.

[115] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[116] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples, 2015.

[117] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[118] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[119] Sven Gowal, Chongli Qin, Po-Sen Huang, Taylan Cemgil, Krishnamurthy Dvijotham, Timothy Mann, and Pushmeet Kohli. Achieving Robustness in the Wild via Adversarial Mixing with Disentangled Representations. *arXiv preprint arXiv:1912.03192*, 2019.

[120] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, and David Duvenaud. Scalable Reversible Generative Models with Free-form Continuous Dynamics. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJxgknCcK7.

[121] Alex Graves. Practical Variational Inference for Neural Networks. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors,

Advances in Neural Information Processing Systems 24, pages 2348–2356. Curran Associates, Inc., 2011. URL http://papers.nips.cc/paper/4329-practical-variational-inference-for-neural-networks.pdf.

[122] Hilary Greaves. Cluelessness. *Proceedings of the Aristotelian Society*, 116(3): 311–339, 2016. doi: 10.1093/arisoc/aow018.

[123] Ishaan Gulrajani and David Lopez-Paz. In Search of Lost Domain Generalization, 2020.

[124] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. *CoRR*, abs/1706.04599, 2017. URL http://arxiv.org/abs/1706.04599.

[125] David Ha, Andrew Dai, and Quoc V. Le. HyperNetworks. In *5th International Conference on Learning Representations*, 2017. URL https://openreview.net/pdf?id=rkpACe1lx.

[126] David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks. *International Conference on Learning Representations*, 2017.

[127] Tuomas Haarnoja, Kristian Hartikainen, Pieter Abbeel, and Sergey Levine. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 1851–1860. PMLR, 2018.

[128] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[129] Dylan Hadfield-Menell, Anca D. Dragan, Pieter Abbeel, and Stuart J. Russell. Cooperative Inverse Reinforcement Learning. *CoRR*, abs/1606.03137, 2016. URL http://arxiv.org/abs/1606.03137.

[130] Patrick Haffner. Escaping the Convex Hull with Extrapolated Vector Machines. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 753–760. MIT Press, 2002.

[131] Moritz Hardt, Benjamin Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*, 2015.

[132] Moritz Hardt, Eric Price, and Nathan Srebro. Equality of Opportunity in Supervised Learning, 2016.

[133] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[134] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *CoRR*, abs/1502.01852, 2015. URL `http://arxiv.org/abs/1502.01852`.

[135] Yue He, Zheyan Shen, and Peng Cui. Towards Non-I.I.D. Image Classification: A Dataset and Baselines, 2019.

[136] Christina Heinze-Deml, Jonas Peters, and Nicolai Meinshausen. Invariant Causal Prediction for Nonlinear Models. *Journal of Causal Inference*, 6(2), Sep 2018. ISSN 2193-3685. doi: 10.1515/jci-2017-0016. URL `http://dx.doi.org/10.1515/jci-2017-0016`.

[137] Dan Hendrycks and Thomas G. Dietterich. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. *CoRR*, abs/1903.12261, 2019. URL `http://arxiv.org/abs/1903.12261`.

[138] Dan Hendrycks and Kevin Gimpel. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. *arXiv preprint arXiv:1610.02136*, 2016.

[139] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[140] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

[141] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty, 2019.

[142] Dan Hendrycks, Norman Mu, Ekin D. Cubuk, Barret Zoph, Justin Gilmer, and Balaji Lakshminarayanan. AugMix: A Simple Data Processing Method to Improve Robustness and Uncertainty, 2019.

[143] Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. Aligning ai with shared human values. *arXiv preprint arXiv:2008.02275*, 2020.

[144] Jose Miguel Hernandez-Lobato and Ryan Adams. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1861–1869, 2015.

[145] R. Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization, 2018.

[146] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[147] Weihua Hu, Gang Niu, Issei Sato, and Masashi Sugiyama. Does Distributionally Robust Supervised Learning Give Robust Classifiers?, 2016.

[148] Chin-Wei Huang, Ahmed Touati, Laurent Dinh, Michal Drozdzal, Mohammad Havaei, Laurent Charlin, and Aaron Courville. Learnable Explicit Density for Con-

tinuous Latent Space and Variational Inference. *arXiv preprint arXiv:1710.02248*, 2017.

[149] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural Autoregressive Flows, 2018.

[150] Xun Huang and Serge J. Belongie. Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. *CoRR*, abs/1703.06868, 2017. URL `http://arxiv.org/abs/1703.06868`.

[151] Evan Hubinger, Chris van Merwijk, Vladimir Mikulik, Joar Skalse, and Scott Garrabrant. Risks from Learned Optimization in Advanced Machine Learning Systems, 2019.

[152] Aapo Hyvärinen and Petteri Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3), 1999.

[153] Maximilian Ilse, Jakub M. Tomczak, and Patrick Forré. Designing Data Augmentation for Simulating Interventions. *arXiv preprint arXiv:2005.01856*, 2020.

[154] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pages 125–136, 2019.

[155] Daniel Jiwoong Im, Michael Tao, and Kristin Branson. An Empirical Analysis of Deep Network Loss Surfaces. *arXiv preprint arXiv:1612.04010*, 2016.

[156] Fredrik D. Johansson, David Sontag, and Rajesh Ranganath. Support and Invertibility in Domain-Invariant Representations, 2019.

[157] Daniel Kahneman. Thinking, Fast and Slow, 2012.

[158] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models, 2020.

[159] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[160] Kaveh Khoshkhah and Dirk Oliver Theis. Fooling Sets and the Spanning Tree Polytope. *CoRR*, abs/1701.00350, 2017. URL `http://arxiv.org/abs/1701.00350`.

[161] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

[162] Diederik P. Kingma and Prafulla Dhariwal. Glow: Generative Flow with Invertible 1x1 Convolutions, 2018.

[163] Diederik P. Kingma and Max Welling. Auto-encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[164] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2014.

[165] Diederik P. Kingma, T. Salimans, and M. Welling. Variational Dropout and the Local Reparameterization Trick. *arXiv e-prints*, June 2015.

[166] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751, 2016.

[167] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, 2016.

[168] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and

Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016. URL http://arxiv.org/abs/1612.00796.

[169] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[170] Pang Wei Koh and Percy Liang. Understanding Black-box Predictions via Influence Functions. *arXiv preprint arXiv:1703.04730*, 2017.

[171] Masanori Koyama and Shoichiro Yamaguchi. Out-of-Distribution Generalization with Maximal Invariant Predictor, 2020.

[172] Victoria Krakovna, Jonathan Uesato, Vladimir Mikulik, Matthew Rahtz, Tom Everitt, Ramana Kumar, Zac Kenton, Jan Leike, and Shane Legg. Specification gaming: the flip side of AI ingenuity, 2020. URL https://deepmind.com/blog/article/Specification-gaming-the-flip-side-of-AI-ingenuity.

[173] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR-10 (Canadian Institute for Advanced Research), 2009. "Learning Multiple Layers of Features from Tiny Images", Alex Krizhevsky, 2009. http://www.cs.toronto.edu/~kriz/cifar.html.

[174] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, volume 25, pages 1097–1105, 2012.

[175] David Krueger. Designing Regularizers and Architectures for Recurrent Neural Networks. Master thesis, Université de Montréal, 2016.

[176] David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*, 2017.

[177] David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Remi Le Priol, and Aaron Courville. Out-of-Distribution Generalization via Risk Extrapolation (REx), 2020.

[178] David Krueger, Jan Leike, Owain Evans, and John Salvatier. Active Reinforcement Learning: Observing Rewards at a Cost. *CoRR*, abs/2011.06709, 2020. URL `https://arxiv.org/abs/2011.06709`.

[179] David Krueger, Tegan Maharaj, and Jan Leike. Hidden Incentives for Auto-Induced Distributional Shift, 2020.

[180] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

[181] Alexandre Lacoste, Thomas Boquet, Negar Rostamzadeh, Boris Oreshkin, Wonchang Chung, and David Krueger. Deep Prior, 2017.

[182] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building Machines That Learn and Think Like People. *CoRR*, abs/1604.00289, 2016. URL `http://arxiv.org/abs/1604.00289`.

[183] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles, 2017.

[184] Hugo Larochelle and Iain Murray. The Neural Autoregressive Distribution Estimator. In *The Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, volume 15 of *JMLR: W&CP*, 2011.

[185] Yann LeCun, Bernhard Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[186] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[187] Yann LeCun, Corinna Cortes, and Christopher J. C. Burges. The MNIST database of handwritten digits, 1998.

[188] Shane Legg and Marcus Hutter. A Collection of Definitions of Intelligence, 2007.

[189] Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A. Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. AI Safety Gridworlds. *CoRR*, abs/1711.09883, 2017. URL `http://arxiv.org/abs/1711.09883`.

[190] Jan Leike, David Krueger, Tom Everitt, Miljan Martic, Vishal Maini, and Shane Legg. Scalable agent alignment via reward modeling: a research direction. *CoRR*, abs/1811.07871, 2018. URL `http://arxiv.org/abs/1811.07871`.

[191] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-End Training of Deep Visuomotor Policies, 2016.

[192] Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M. Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pages 5542–5550, 2017.

[193] Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 624–639, 2018.

[194] Yingzhen Li and Yarin Gal. Dropout Inference in Bayesian Neural Networks with Alpha-divergences. *arXiv preprint arXiv:1703.02914*, 2017.

[195] M. Lichman. UCI Machine Learning Repository, 2013. URL `http://archive.ics.uci.edu/ml`.

[196] Falk Lieder and Thomas L. Griffiths. Resource-rational analysis: Understanding human cognition as the optimal use of limited computational resources. *Behavioral and Brain Sciences*, 43:e1, 2020. doi: 10.1017/S0140525X1900061X.

[197] Henry W. Lin and Max Tegmark. Why does deep and cheap learning work so well? *arXiv preprint arXiv:1608.08225*, 2016.

[198] Zachary C. Lipton, Yu-Xiang Wang, and Alex Smola. Detecting and correcting for label shift with black box predictors. *arXiv preprint arXiv:1802.03916*, 2018.

[199] Anqi Liu, Lev Reyzin, and Brian Ziebart. Shift-pessimistic active learning using robust bias-aware prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[200] Gabriel Loaiza-Ganem, Yuanjun Gao, and John P. Cunningham. Maximum entropy flow networks. In *International Conference on Learning Representations*, 2017.

[201] Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I. Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pages 1640–1650, 2018.

[202] Sarah Loos, Geoffrey Irving, Christian Szegedy, and Cezary Kaliszyk. Deep Network Guided Proof Search, 2017.

[203] Christos Louizos and Max Welling. Multiplicative Normalizing Flows for Variational Bayesian Neural Networks. *arXiv e-prints*, March 2017.

[204] Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. In *International Conference on Machine Learning*, 2017.

[205] Andrew L. Maas, Awni Y. Hannun, Andrew Y. Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28, 2013.

[206] Michael MacAskill, Krister Bykvist, and Toby Ord. *Moral uncertainty*. Oxford University Press, 2020.

[207] David J. C. MacKay. Bayesian Neural Networks and Density Networks. In *Nuclear Instruments and Methods in Physics Research, A.*, pages 73–80, 1994.

[208] Dougal Maclaurin, David K. Duvenaud, and Ryan P. Adams. Gradient-based Hyperparameter Optimization through Reversible Learning. In *ICML*, pages 2113–2122, 2015.

[209] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In *32nd Conference on Neural Information Processing Systems (NeurIPS 2018), Montréal, Canada*, 2018.

[210] David Manheim and Scott Garrabrant. Categorizing Variants of Goodhart's Law. *CoRR*, abs/1803.04585, 2018. URL `http://arxiv.org/abs/1803.04585`.

[211] David Martin, Charless Fowlkes, Doron Tal, and Jitendra Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2. IEEE, 2001.

[212] Nicolai Meinshausen, Peter Bühlmann, et al. Maximin effects in inhomogeneous large-scale data. *The Annals of Statistics*, 43(4):1801–1830, 2015.

[213] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. Distributional Smoothing with Virtual Adversarial Training. *stat*, 1050:25, 2015.

[214] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing Atari with Deep Reinforcement Learning. *CoRR*, abs/1312.5602, 2013. URL `http://arxiv.org/abs/1312.5602`.

[215] Grégoire Montavon, Mikio L. Braun, and Klaus-Robert Müller. Kernel Analysis of Deep Networks. *Journal of Machine Learning Research*, 12, 2011.

[216] Guido F. Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the Number of Linear Regions of Deep Neural Networks. In Z. Ghahramani,

M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2924–2932. Curran Associates, Inc., 2014.

[217] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. Inceptionism: Going Deeper into Neural Networks. Google AI Blog, 2015. `https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html`.

[218] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. Deep Learning Recommendation Model for Personalization and Recommendation Systems, 2019.

[219] Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996. ISBN 0387947248.

[220] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.

[221] Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pages 278–287, 1999.

[222] Richard Ngo. AGI Safety From First Principles, 2020. `https://drive.google.com/file/d/1uK7NhdSKprQKZnRjU58X7NLA1auXlWHt/view`.

[223] Srivastava Nitish, Hinton Geoffrey, Krizhevsky Alex, Sutskever Ilya, and Salakhutdinov Ruslan. Dropout: A Simple Way to Prevent Neural Networks from Over-

fitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2627435.2670313.

[224] Junier B. Oliva, Avinava Dubey, Barnabás Póczos, Jeff Schneider, and Eric P. Xing. Transformation Autoregressive Networks. *arXiv preprint arXiv:1801.09819*, 2018.

[225] Stephen M. Omohundro. The basic AI drives, 2008.

[226] Toby Ord. *The Precipice*. Bloomsbury Publishing, 2020.

[227] Laurent Orseau and M. S. Armstrong. Safely interruptible agents. In Alexander Ihler and Dominik Janzing, editors, *UAI'16: Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pages 557–566. Association for Uncertainty in Artificial Intelligence, 2016.

[228] Pedro A. Ortega, Vishal Maini, and the DeepMind safety team. Building safe artificial intelligence: specification, robustness, and assurance. DeepMind Safety Research, 2018. https://deepmindsafetyresearch.medium.com/building-safe-artificial-intelligence-52f5f75058f1.

[229] Sinno Jialin Pan, Ivor W. Tsang, James T. Kwok, and Qiang Yang. Domain adaptation via transfer component analysis. *IEEE Transactions on Neural Networks*, 22 (2):199–210, 2010.

[230] George Papamakarios, Iain Murray, and Theo Pavlakou. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, 2017.

[231] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[232] Ethan Perez, Harm de Vries, Florian Strub, Vincent Dumoulin, and Aaron C. Courville. Learning Visual Reasoning Without Strong Priors. *CoRR*, abs/1707.03017, 2017. URL http://arxiv.org/abs/1707.03017.

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

[234] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

[235] Stanislas Polu and Ilya Sutskever. Generative Language Modeling for Automated Theorem Proving, 2020.

[236] Ben Poole, Subhaneil Lahiri, Maithreyi Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 3360–3368. Curran Associates, Inc., 2016.

[237] Arya A. Pourzanjani, Richard M. Jiang, and Linda R. Petzold. Improving the identifiability of neural networks for Bayesian inference. In *NIPS Workshop on Bayesian Deep Learning*, volume 4, page 29, 2017.

[238] Powercube. Foucault: power is everywhere, 2021. https://www.powercube.net/other-forms-of-power/foucault-power-is-everywhere/.

[239] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. *arXiv preprint arXiv:1606.05336*, 2016.

[240] Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. CNN Features off-the-shelf: an Astounding Baseline for Recognition. *CoRR*, abs/1403.6382, 2014. URL `http://arxiv.org/abs/1403.6382`.

[241] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do ImageNet Classifiers Generalize to ImageNet? *arXiv preprint arXiv:1902.10811*, 2019.

[242] Laria Reynolds and Kyle McDonell. Prompt programming for large language models: Beyond the few-shot paradigm. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–7, 2021.

[243] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1530–1538, 2015.

[244] Danilo Jimenez Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, 2015.

[245] Danilo Jimenez Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *arXiv e-prints*, January 2014.

[246] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, 2014.

[247] Oren Rippel and Ryan Prescott Adams. High-dimensional probability estimation with deep density models. *arXiv preprint arXiv:1302.5125*, 2013.

[248] Stuart Russell. *Human Compatible: Artificial Intelligence and the Problem of Control*. Viking Press, 2019.

[249] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach, Global Edition*. Pearson, 3 edition, 2016.

[250] Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization, 2019.

[251] Subham S. Sahoo, Christoph H. Lampert, and Georg Martius. Learning Equations for Extrapolation and Control, 2018.

[252] Ananya B. Sai, Akash Kumar Mohankumar, and Mitesh M. Khapra. A Survey of Evaluation Metrics Used for NLG Systems, 2020.

[253] Tim Salimans and Diederik P. Kingma. Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks. *CoRR*, abs/1602.07868, 2016. URL `http://arxiv.org/abs/1602.07868`.

[254] Tim Salimans, Diederik P. Kingma, and Max Welling. Markov chain Monte Carlo and variational inference: Bridging the gap. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1218–1226, 2015.

[255] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[256] Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. On Causal and Anticausal Learning. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pages 459–466, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851.

[257] Lukas Schott, Julius von Kügelgen, Frederik Träuble, Peter V. Gehler, Chris Russell, Matthias Bethge, Bernhard Schölkopf, Francesco Locatello, and Wieland Brendel. Visual Representation Learning Does Not Generalize Strongly Within the Same Domain. *CoRR*, abs/2107.08221, 2021. URL `https://arxiv.org/abs/2107.08221`.

[258] Rohin Shah, Pedro Freire, Neel Alex, Rachel Freedman, Dmitrii Krasheninnikov, Lawrence Chan, Michael D. Dennis, Pieter Abbeel, Anca Dragan, and Stuart

Russell. Benefits of Assistance over Reward Learning, 2021. URL shah2021benefits.

[259] Jiaxin Shi, Shengyang Sun, and Jun Zhu. Implicit Variational Inference with Kernel Density Ratio Fitting. *arXiv preprint arXiv:1705.10119*, 2017.

[260] Connor Shorten and Taghi M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[261] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676): 354–359, 2017.

[262] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy Lillicrap, Karen Simonyan, and Demis Hassabis. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[263] Karl Sims. Evolving virtual creatures. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 15–22, 1994.

[264] Aman Sinha, Hongseok Namkoong, Riccardo Volpi, and John Duchi. Certifying Some Distributional Robustness with Principled Adversarial Training, 2017.

[265] Walter Sinnott-Armstrong. Consequentialism: "Consequences of What? Rights, Relativity, and Rules". In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, Fall 2021 edition, 2021.

[266] J. Sjoberg, J. Sjoeberg, J. Sjöberg, and L. Ljung. Overtraining, regularization and searching for a minimum, with application to neural networks. *International Journal of Control*, 62:1391–1407, 1995.

[267] Nate Soares, Benja Fallenstein, Stuart Armstrong, and Eliezer Yudkowsky. Corrigibility. In *Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[268] Jure Sokolic, Raja Giryes, Guillermo Sapiro, and Miguel R. D. Rodrigues. Robust large margin deep neural networks. *arXiv preprint arXiv:1605.08254*, 2016.

[269] Jiaming Song, Shengjia Zhao, and Stefano Ermon. A-nice-mc: Adversarial training for mcmc. In *Advances in Neural Information Processing Systems*, 2017.

[270] Xingyi Song, Trevor Cohn, and Lucia Specia. BLEU deconstructed: Designing a better MT evaluation metric. *International Journal of Computational Linguistics and Applications*, 4(2):29–44, 2013.

[271] Daniel Soudry, Itay Hubara, and Ron Meir. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In *Advances in Neural Information Processing Systems*, pages 963–971, 2014.

[272] Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F. Christiano. Learning to summarize from human feedback. *CoRR*, abs/2009.01325, 2020. URL `https://arxiv.org/abs/2009.01325`.

[273] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

[274] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[275] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013. URL `http://arxiv.org/abs/1312.6199`.

[276] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[277] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2014.

[278] Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind Control Suite. Technical report, DeepMind, January 2018.

[279] Jessica Taylor. On motivations for MIRI's highly reliable agent design research. LessWrong, 2017. `https://www.lesswrong.com/posts/5bd75cc58225bf0670375321/on-motivations-for-miri-s-highly-reliable-agent-design-research#The_concern`.

[280] Max Tegmark. *Life 3.0: Being human in the age of artificial intelligence*. Knopf, 2017.

[281] Gerald Tesauro et al. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

[282] Theano Development Team et al. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[283] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive Multiview Coding, 2019.

[284] Jakub M. Tomczak and Max Welling. Improving variational auto-encoders using householder flow. *arXiv preprint arXiv:1611.09630*, 2016.

[285] Jakub M. Tomczak and Max Welling. Improving Variational Auto-Encoders using convex combination linear Inverse Autoregressive Flow. In *Benelearn*, 2017.

[286] Antonio Torralba and Alexei A. Efros. Unbiased look at dataset bias. In *CVPR 2011*, pages 1521–1528. IEEE, 2011.

[287] Alan Turing. Intelligent Machinery, A Heretical Theory (c.1951). In B. Jack Copeland, editor, *The Essential Turing: Seminal Writings in Computing, Logic, Philosophy, Artificial Intelligence, and Artificial Life: Plus The Secrets of Enigma*, chapter 12. Oxford University Press, 2004. http://www.cse.chalmers.se/~aikmitr/papers/Turing.pdf#page=474.

[288] Alexander Matt Turner, Logan Smith, Rohin Shah, Andrew Critch, and Prasad Tadepalli. Optimal policies tend to seek power, 2021.

[289] Richard E. Turner and Maneesh Sahani. Two problems with variational expectation maximisation for time-series models. *Bayesian Time series models*, 1(3.1), 2011.

[290] Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7167–7176, 2017.

[291] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance Normalization: The Missing Ingredient for Fast Stylization. *CoRR*, abs/1607.08022, 2016. URL http://arxiv.org/abs/1607.08022.

[292] Benigno Uria, Iain Murray, and Hugo Larochelle. RNADE: The real-valued neural autoregressive density-estimator. In *Advances in Neural Information Processing Systems*, 2013.

[293] Aaron van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *ArXiv e-prints*, September 2016.

[294] Aaron van den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International Conference on Machine Learning*, 2016.

[295] Aaron van den Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George van den Driessche, Edward Lockhart, Luis C. Cobo, Florian Stimberg, et al. Parallel WaveNet: Fast High-Fidelity Speech Synthesis. *arXiv preprint arXiv:1711.10433*, 2017.

[296] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation Learning with Contrastive Predictive Coding, 2018.

[297] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[298] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need, 2017.

[299] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[300] Yoav Wald, Amir Feder, Daniel Greenfeld, and Uri Shalit. On Calibration and Out-of-domain Generalization. *CoRR*, abs/2102.10395, 2021. URL `https://arxiv.org/abs/2102.10395`.

[301] Haohan Wang, Zexue He, Zachary C. Lipton, and Eric P. Xing. Learning robust representations by projecting superficial statistics out. *arXiv preprint arXiv:1903.06256*, 2019.

[302] Shengjie Wang, Abdel-rahman Mohamed, Rich Caruana, Jeff Bilmes, Matthai Plilipose, Matthew Richardson, Krzysztof Geras, Gregor Urban, and Ozlem Aslan. Analysis of Deep Neural Networks with the Extended Data Jacobian Matrix. In *International Conference on Machine Learning*, pages 718–726. PMLR, 2016.

[303] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned Language Models Are Zero-Shot Learners, 2021.

[304] Max Welling and Yee W. Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

[305] Max Welling and Yee Whye Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics, 2011.

[306] Ben West. Problems and Solutions in Infinite Ethics. Effective Altruism Forum, 2015. https://forum.effectivealtruism.org/posts/9D6zKRPfaALiBhnnN/problems-and-solutions-in-infinite-ethics.

[307] Norbert Wiener. Some moral and technical consequences of automation. *Science*, 131(3410):1355–1358, 1960. https://pdfs.semanticscholar.org/8a40/3ab09db56252fb538dfd95509472661c6eb6.pdf.

[308] Wikipedia contributors. Goodhart's law — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Goodhart%27s_law&oldid=1054752097, 2021.

[309] Robert C. Williamson and Aditya Krishna Menon. Fairness risk measures, 2019.

[310] D. Randall Wilson and Tony R. Martinez. The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10):1429–1451, 2003.

[311] Xi Wu, Yang Guo, Jiefeng Chen, Yingyu Liang, Somesh Jha, and Prasad Chalasani. Representation Bayesian Risk Decompositions and Multi-Source Domain Adaptation, 2020.

[312] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[313] Yazhou Yang and Marco Loog. A benchmark and comparison of active learning for logistic regression. *Pattern Recognition*, 83:401–415, 2018. ISSN 0031-3203. doi: https://doi.org/10.1016/j.patcog.2018.06.004. URL https://www.sciencedirect.com/science/article/pii/S0031320318302140.

[314] Yuan Yao, Lorenzo Rosasco, and Andrea Caponnetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

[315] Eliezer Yudkowsky. Coherent extrapolated volition. *Singularity Institute for Artificial Intelligence*, 2004.

[316] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

[317] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations (ICLR)*, 2017.

[318] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond Empirical Risk Minimization, 2017.

[319] Han Zhao, Remi Tachet des Combes, Kun Zhang, and Geoffrey J. Gordon. On Learning Invariant Representation for Domain Adaptation, 2019.

[320] Liang Zhou, Chin-Yew Lin, and Eduard Hovy. Re-evaluating machine translation results with paraphrase support. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 77–84, 2006.

CHAPTER 11

APPENDIX

11.1 Appendix for "Out-of-Distribution Generalization via Risk Extrapolation (REx)"

11.1.1 Appendix overview

Our code is available online at: https://anonymous.4open.science/r/12747e81-8505-43cb-b54e-e75e2344a397/. The sections of our appendix are as follows:

A) Appendix Overview

B) Definition and discussion of extrapolation in machine learning

C) Illustrative examples of how REx works in toy settings

D) A summary of different types of causal model

E) Theory

F) The relationship between MM-REx vs. V-REx, and the role each plays in our work

G) Further results and details for experiments mentioned in main text

H) Experiments not mentioned in main text

I) Overview of other topics related to OOD generalization

11.1.2 Definition and discussion of extrapolation in machine learning

We define interpolation and extrapolation as follows: **interpolation** refers to making decisions or predictions about points *within* the convex hull of the training examples and **extrapolation** refers to making decisions or predictions about points *outside* their convex

hull.[1] This generalizes the familiar sense of these terms for one-dimensional functions. An interesting consequence of this definition is: for data of high intrinsic dimension, generalization *requires* extrapolation [133], even in the i.i.d. setting. This is because the volume of high-dimensional manifolds concentrates near their boundary; see Figure 11.1.

11.1.2.1 Extrapolation in the space of risk functions

The same geometric considerations apply to extrapolating to new domains. Domains can be highly diverse, varying according to high dimensional attributes, and thus requiring extrapolation to generalize across. Thus Risk Extrapolation might often do a better job of including possible test domains in its perturbation set than Risk Interpolation does.

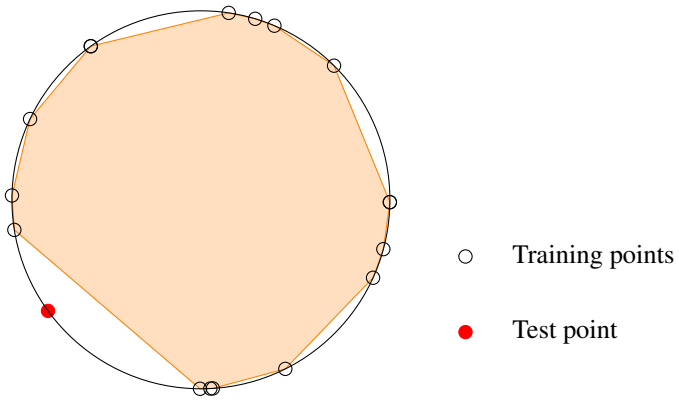


Figure 11.1: Illustration of the importance of extrapolation for generalizing in high dimensional space. In high dimensional spaces, mass concentrates near the boundary of objects. For instance, the uniform distribution over a ball in $N+1$-dimensional space can be approximated by the uniform distribution over the N-dimensional hypersphere. We illustrate this in 2 dimensions, using the 1-sphere (i.e. the unit circle). Dots represent a finite training sample, and the shaded region represents the convex hull of all but one member of the sample. Even in 2 dimensions, we can see why any point from a finite sample from such a distribution remains outside the convex hull of the other samples, with probability 1. The only exception would be if two points in the sample coincide *exactly*.

[1] Surprisingly, we were not able to find any existing definition of these terms in the machine learning literature. They have been used in this sense [130, 133], but also to refer to strong generalization capabilities more generally [251].

11.1.3 Illustrative examples of how REx works in toy settings

Here, we work through two examples to illustrate:

1. How to understand extrapolation in the space of probability density/mass functions (PDF/PMFs)

2. How REx encourages robustness to covariate shift via distributing capacity more evenly across possible input distributions.

11.1.4 6D example of REx

Here we provide a simple example illustrating how to understand extrapolations of probability distributions. Suppose $X \in \{0,1,2\}$ and $Y \in \{0,1\}$, so there are a total of 6 possible types of examples, and we can represent their distributions in a particular domain as a point in 6D space: $(P(0,0),P(0,1),P(1,0),P(1,1),P(2,0),P(2,1))$. Now, consider three domains e_1, e_2, e_3 given by

1. (a,b,c,d,e,f)

2. $(a,b,c,d,e-k,f+k)$

3. $(2a,2b,c(1-\frac{a+b}{c+d}),d(1-\frac{a+b}{c+d}),e,f)$

The difference between e_1 and e_2 corresponds to a shift in $P(Y|X=2)$, and suggests that Y cannot be reliably predicted across different domains when $X=2$. Meanwhile, the difference between e_1 and e_3 tells us that the relative probability of $X=0$ vs. $X=1$ can change, and so we might want our model to be robust to these sorts of covariate shifts. Extrapolating risks across these 3 domains effectively tells the model: "don't bother trying to predict Y when $X=2$ (i.e. aim for $\hat{P}(Y=1|X=2)=.5$), and split your capacity equally across the $X=0$ and $X=1$ cases". By way of comparison, IRM would also aim for $\hat{P}(Y=1|X=2)=.5$, whereas ERM would aim for $\hat{P}(Y=1|X=2)=\frac{3f+k}{3e+3f}$ (assuming $|D_1|=|D_2|=|D_3|$). And unlike REx, both ERM and IRM would split capacity between $X=0/1/2$ cases according to their empirical frequencies.

11.1.5 Covariate shift example

We now give an example to show how REx provides robustness to covariate shift. Covariate shift is an issue when a model has limited capacity or limited data.

Viewing REx as robust learning over the affine span of the training distributions reveals its potential to improve robustness to distribution shifts. Consider a situation in which a model encounters two types of inputs: COSTLY inputs with probability q and CHEAP inputs with probability $1-q$. The model tries to predicts the input – it outputs COSTLY with probability p and CHEAP with probability $1-p$. If the model predicts right its risk is 0, but if it predicts COSTLY instead of CHEAP it gets a risk $u=2$, and if it predicts CHEAP instead of COSTLY it gets a risk $v=4$. The risk has expectation $\mathbb{R}_q(p) = (1-p)(1-q)u + pqv$. We have access to two domains with different input probabilities $q_1 < q_2$. This is an example of pure covariate shift.

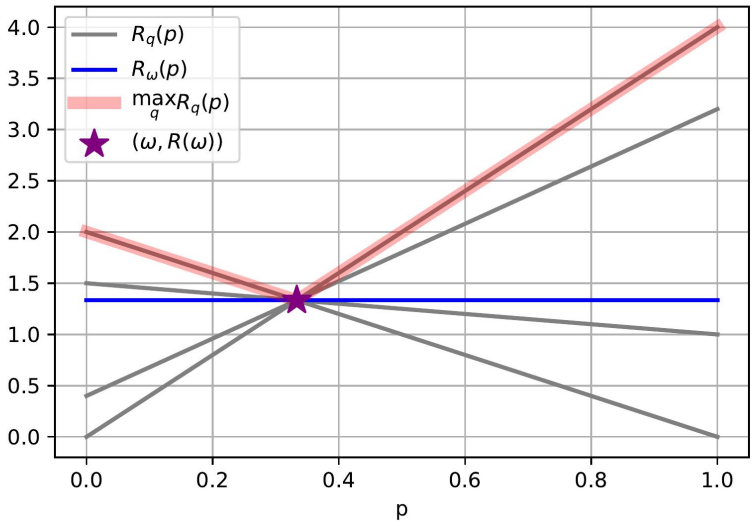


Figure 11.2: Each grey line is a risk $\mathbb{R}_q(p)$ as functions of p for a specific value of q. The blue line is when $q=\omega$. We highlight in red the curve $\max_q \mathbb{R}_q(p)$ whose minimum is the saddle point marked by a purple star in $p=\omega$.

We want to guarantee the minimal risk over the set of all possible domains:

$$\min_{p\in[0,1]} \max_{q\in[0,1]} \mathbb{R}_q(p) = (1-p)(1-q)u + pqv$$

as illustrated in Figure 11.2. The saddle point solution of this problem is $p = \omega = u/u+v$ and $\mathbb{R}_q(p) = uv/u+v, \forall q$. From the figure we see that $\mathbb{R}_{q_1}(p) = \mathbb{R}_{q_2}(p)$ can only happen for $p = \omega$, so the risk extrapolation principle will return the minimax optimal solution.

If we use ERM to minimize the risk, we will pool together the domains into a new domain with COSTLY input probability $\bar{q} = (q_1 + q_2)/2$. ERM will return $p = 0$ if $\bar{q} > \omega$ and $p = 1$ otherwise. Risk interpolation (RI) $\min_p \max_{q\in\{q_1,q_2\}} R_q(p)$ will predict $p = 0$ if $q_1, q_2 > \omega$, $p = 1$ if $q_1, q_2 < \omega$ and $p = \omega$ if $q_1 < \omega < q_2$. We see that only REx finds the minimax optimum for arbitrary values of q_1 and q_2.

11.1.6 A summary of different types of causal models

Here, we briefly summarize the differences between 3 different types of causal models, see Table 11.I. Our definitions and notation follow *Elements of Causal Inference: Foundations and Learning Algorithms* [234].

A **Causal Graph** is a directed acyclic graph (DAG) over a set of nodes corresponding to random variables $\mathbf{Z}$, where edges point from causes (including noise variables) to effects. A **Structural Causal Model (SCM)**, $\mathfrak{C}$, additionally specifies a *deterministic* mapping f_Z for every node Z, which computes the value of that node given the values of its parents, which include a special noise variable N_Z, which is sampled independently from all other nodes. This f_Z is called the **mechanism**, **structural equation**, or **structural assignment** for Z. Given an SCM, $\mathfrak{C}$, the **entailed distribution** of $\mathfrak{C}$, $P^{\mathfrak{C}}(\mathbf{Z})$ is defined via ancestral sampling. Thus for any $Z \in \mathbf{Z}$, we have that the marginal distribution $P^{\mathfrak{C}}(Z|\mathbf{Z} \setminus Z) = P^{\mathfrak{C}}(Z|Pa(Z))$. A **Causal Graphical Model (CGM)** can be thought of as specifying these marginal distributions *without* explicitly representing noise variables N_Z. We can draw rough analogies with (non-causal) statistical models. Roughly speaking, Causal Graphs are analogous to Graphical Models, whereas SCMs and CGMs are analogous to joint distributions.

Model	Independences	Distributions	Interventions	Counterfactuals
Graphical Model	✓	✗	✗	✗
Joint Distribution	✓	✓	✗	✗
Causal Graph	✓	✗	✓	✗
Causal Graphical Model	✓	✓	✓	✗
Structural Causal Model	✓	✓	✓	✓

Table 11.I: A comparison of causal and non-causal models.

11.1.7 Theory

11.1.7.1 Proofs of theorems 1 and 2

The REx principle has two goals:

1. Reducing training risks
2. Increasing similarity of training risks.

In practice, it may be advantageous to trade-off these two objectives, using a hyperparameter (e.g. β for V-REx or λ_{min} for MM-REx). However, in this section, we assume the 2nd criteria takes priority; i.e. we define "satisfying" the REx principle as selecting a minimal risk predictor among those that achieve *exact* equality of risks across all the domains in a set $\mathscr{E}$.

Recall our assumptions from Section 5.3.4 of the main text:

1. The causes of Y are observed, i.e. $Pa(Y) \subseteq X$.
2. Domains correspond to interventions on X.
3. Homoskedasticity (a slight generalization of the additive noise setting assumed by Peters et al. [233]). We say an SEM $\mathfrak{C}$ is **homoskedastic** (with respect to a loss function ℓ), if the Bayes error rate of $\ell(f_Y(x), f_Y(x))$ is the same for all $x \in \mathscr{X}$.

And see Section 5.2.3 for relevant definitions and notation.

We begin with a theorem based on the setting explored by Peters et al. [233]. Here, $\varepsilon_i \doteq N_i$ are assumed to be normally distributed.

Theorem 1. *Given a Linear SEM, $X_i \leftarrow \sum_{j \neq i} \beta_{(i,j)} X_j + \varepsilon_i$, with $Y \doteq X_0$, and a predictor $f_\beta(X) \doteq \sum_{j:j>0} \beta_j X_j + \varepsilon_j$ that satisfies REx (with mean-squared error) over a perturbation set of domains that contains 3 distinct $do()$ interventions for each $X_i : i > 0$. Then $\beta_j = \beta_{0,j}, \forall j$.*

Proof. We adapt the proof of Theorem 4i from Peters et al. [233] to show that REx will learn the correct model under similar assumptions. Let $Y \leftarrow \gamma X + \varepsilon$ be the mechanism for Y, assumed to be fixed across all domains, and let $\hat{Y} = \beta X$ be our predictor. Then the residual is $R(\beta) = (\gamma - \beta)X + \varepsilon$. Define $\alpha_i \doteq \gamma_i - \beta_i$, and consider an intervention $do(X_j = x)$ on the youngest node X_j with $\alpha_j \neq 0$. Then as in eqn 36/37 of Peters et al. [233], we compare the residuals R of this intervention and of the observational distribution:

$$R^{\text{obs}}(\beta) = \alpha_j X_j + \sum_{i \neq j} \alpha_i X_i + \varepsilon \qquad R^{do(X_j = x)}(\beta) = \alpha_j x + \sum_{i \neq j} \alpha_i X_i + \varepsilon \tag{11.1}$$

We now compute the MSE risk for both domains, set them equal, and simplify to find a quadratic formula for x:

$$\mathbb{E}\left[(\alpha_j X_j + \sum_{i \neq j} \alpha_i X_i + \varepsilon)^2\right] = \mathbb{E}\left[(\alpha_j x + \sum_{i \neq j} \alpha_i X_i + \varepsilon)^2\right] \tag{11.2}$$

$$0 = \alpha_j^2 x^2 + 2\alpha_j \mathbb{E}[\sum_{i \neq j} \alpha_i X_i + \varepsilon] x - \mathbb{E}\left[(\alpha_j X_j)^2 - 2\alpha_j X_j (\sum_{i \neq j} \alpha_i X_i + \varepsilon)\right] \tag{11.3}$$

Since there are at most two values of x that satisfy this equation, any other value leads to a violation of REx, so that α_j needs to be zero – contradiction. In particular having domains with 3 different *do*-interventions on every X_i guarantees that the risks are not equal across all domains. □

Given the assumption that a predictor satisfies REx over *all* interventions that do not change the mechanism of Y, we can prove a much more general result. We now consider

an arbitrary SCM, $\mathfrak{C}$, generating Y and X, and let $\mathscr{E}^I$ be the set of domains corresponding to arbitrary interventions on X, similarly to Peters et al. [233].

We emphasize that the predictor is not restricted to any particular class of models, and is a generic function $f : \mathscr{X} \to \mathscr{P}(Y)$, where $\mathscr{P}(Y)$ is the set of distributions over Y. Hence, we drop θ from the below discussion and simply use f to represent the predictor, and $\mathbb{R}(f)$ its risk.

Theorem 2. *Suppose ℓ is a (strictly) proper scoring rule. Then a predictor that satisfies REx for a over $\mathscr{E}^I$ uses $f_Y(x)$ as its predictive distribution on input x for all $x \in \mathscr{X}$.*

Proof. Let $\mathbb{R}^e(f,x)$ be the loss of predictor f on point x in domain e, and $\mathbb{R}^e(f) = \int_{P^e(x)} \mathbb{R}^e(f,x)$ be the risk of f in e. Define $\iota(x)$ as the domain given by the intervention $do(X = x)$, and note that $\mathbb{R}^{\iota(x)}(f) = \mathbb{R}^{\iota(x)}(f,x)$. We additionally define $X_1 \doteq Par(Y)$.

The causal mechanism, f_Y, satisfies the REx principle over $\mathscr{E}^I$. For every $x \in \mathscr{X}$, $f_Y(x) = P(Y|do(X = x)) = P(Y|do(X_1 = x_1)) = P(Y|X_1 = x_1)$ is invariant (meaning 'independent of domain') by definition; $P(Y|do(X = x)) = P(Y|do(X_1 = x_1)) = P(Y|X_1 = x_1)$ follows from the semantics of SEM/SCMs, and the fact that we don't allow f_Y to change across domains. Specifically Y is always generated by the same ancestral sampling process that only depends on X_1 and N^Y. Thus the risk of the predictor $f_Y(x)$ at point x, $\mathbb{R}^e(f_Y,x) = \ell(f_Y(x), f_Y(x))$ is also invariant, soit $\mathbb{R}(f_Y,x)$. Thus $\mathbb{R}^e(f_Y) = \int_{P^e(x)} \mathbb{R}^e(f_Y,x) = \int_{P^e(x)} \mathbb{R}(f_Y,x)$ is invariant whenever $\mathbb{R}(f_Y,x)$ does not depend on x, and the homoskedasticity assumption ensures that this is the case. This establishes that setting $f = f_Y$ will produce equal risk across domains.

No other predictor satisfies the REx principle over $\mathscr{E}^I$. We show that any other g achieves higher risk than f_Y for at least one domain. This demonstrates both that f_Y achieves minimal risk (thus satisfying REx), and that it is the unique predictor which does so (and thus no other predictors satisfy REx). We suppose such a g exists and construct an domain where it achieves higher risk than f_Y. Specifically, if $g \neq f_Y$ then let $x \in \mathscr{X}$ be a point such that $g(x) \neq f_Y(x)$. And since ℓ is a strictly proper scoring rule, this implies that $\ell(g(x), f_Y(x)) > \ell(f_Y(x), f_Y(x))$. But $\ell(g(x), f_Y(x))$ is exactly the risk of g on the domain $\iota(do(X = x))$, and thus g achieves higher risk than f_Y in $\iota(do(X = x))$,

a contradiction. □

11.1.7.2 REx as DRO

We note that MM-REx is also performing robust optimization over a convex hull, see Figure 5.1. The corners of this convex hull correspond to "extrapolated domains" with coefficients $(\lambda_{\min}, \lambda_{\min}, ..., (1-(m-1)\lambda_{\min}))$ (up to some permutation). However, these domains do not necessarily correspond to valid probability distributions; in general, they are quasidistributions, which can assign negative probabilities to some examples. This means that, even if the original risk functions were convex, the extrapolated risks need not be. However, in the case where they *are* convex, then existing theorems, such as the convergence rate result of [250]. This raises several important questions:

1. When is the affine combination of risks convex?
2. What are the effects of negative probabilities on the optimization problem REx faces, and the solutions ultimately found?

11.1.7.3 Negative probabilities:

Figure 11.3 illustrates this for a case where $\mathscr{X} = \mathbb{Z}_2^2$, i.e. x is a binary vector of length 2. Suppose x_1, x_2 are independent in our training domains, and represent the distribution for a particular domain by the point $(P(X_1 = 1), P(X_2 = 1))$. And suppose our 4 training distributions have $(P(X_1 = 1), P(X_2 = 1))$ equal to $\{(.4,.1),(.4,.9),(.6,.1),(.6,.9)\}$, with $P(Y|X)$ fixed.

11.1.8 The relationship between MM-REx vs. V-REx, and the role each plays in our work

The MM-REx and V-REx methods play different roles in our work:

- We use MM-REx to illustrate that REx can be instantiated as a variant of robust optimization, specifically a generalization of the common Risk Interpolation approach. We also find MM-REx provides a useful geometric intuition, since we can

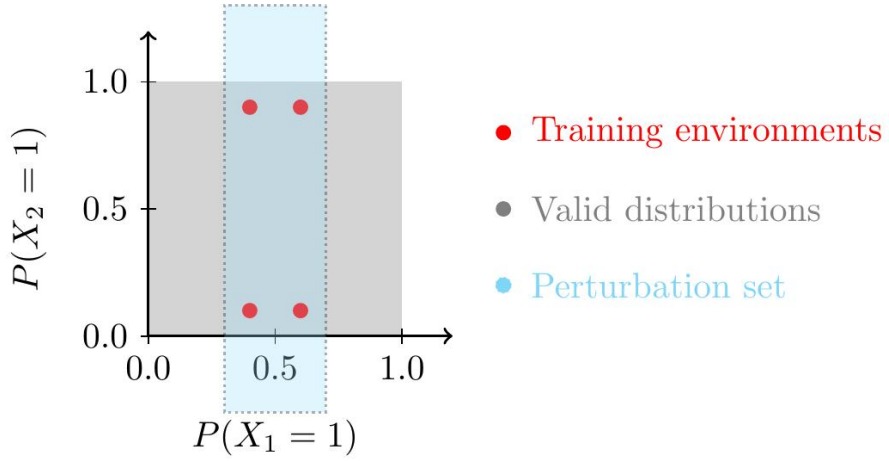


Figure 11.3: The perturbation set for MM-REx can include "distributions" which assign invalid (e.g. negative) probabilities to some data-points. The range of valid distributions $P(X)$ is shown in grey, and $P(X)$ for 4 different training domains are shown as red points. The interior of the dashed line shows the perturbation set for $\lambda_{\text{min}} = -1/2$.

visualize its perturbation set as an expansion of the convex hull of the training risks or distributions.

- We expect V-REx to be the more practical algorithm. It is simple to implement. And it performed better in our CMNIST experiments; we believe this may be due to V-REx providing a smoother gradient vector field, and thus more stable optimization, see Figure 11.1.8.

Either method recovers the REx principle as a limiting case, as we prove in Section 11.1.8.1. We also provide a sequence of mathematical derivations that sheds light on the relationship between MM-REx and V-REx in Section 11.1.8.2 we can view these as a progression of steps for moving from the robust optimization formulation of MM-REx to the penalty term of V-REx:

1. **From minimax to closed form:** We show how to arrive at the closed-form version of MM-REx provided in Eqn. 5.7.

2. **Closed form as mean absolute error:** The closed form of MM-REx is equivalent to a mean absolute error (MAE) penalty term when there are only two training domains.

3. **V-REx as mean squared error:** V-REx is exactly equivalent to a mean squared error penalty term (always). Thus in the case of only two training domains, the difference between MM-REx and V-REx is just a different choice of norm.

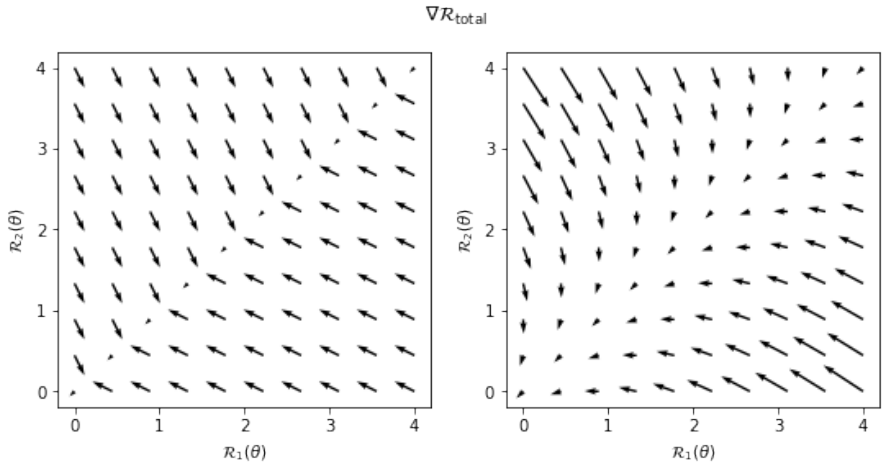


Figure 11.4: Vector fields of the gradient evaluated at different values of training risks $\mathbb{R}_1(\theta)$, $\mathbb{R}_2(\theta)$. We compare the gradients for $\mathbb{R}_{\text{MM-REx}}$ (**left**) and $\mathbb{R}_{\text{V-REx}}$ (**right**). Note that for $\mathbb{R}_{\text{V-REx}}$, the gradient vectors curve smoothly towards the direction of the origin, as they approach the diagonal (where training risks are equal); this leads to a smoother optimization landscape.

11.1.8.1 V-REx and MM-REx enforce the REx principle in the limit

We prove that both MM-REx and V-REx recover the constraint of perfect equality between risks in the limit of $\lambda_{\text{min}} \to -\infty$ or $\beta \to \infty$, respectively. For both proofs, we assume all training risks are finite.

Proposition 4. *The MM-REx risk of predictor* f_θ, $\mathbb{R}_{\text{MM-REx}}(\theta) \to \infty$ *as* $\lambda_{\text{min}} \to -\infty$ *unless* $\mathbb{R}^d = \mathbb{R}^e$ *for all training domains* d, e.

Proof. Suppose the risk is not equal across domains, and let the largest difference between any two training risks be $\varepsilon > 0$. Then $\mathbb{R}_{\text{MM-REx}}(\theta) = (1 - m\lambda_{\text{min}}) \max_e \mathbb{R}_e(\theta) + \lambda_{\text{min}} \sum_{i=1}^m \mathbb{R}_i(\theta) = \max_e \mathbb{R}_e(\theta) - m\lambda_{\text{min}} \max_e \mathbb{R}_e(\theta) + \lambda_{\text{min}} \sum_{i=1}^m \mathbb{R}_i(\theta) \geq \max_e \mathbb{R}_e(\theta) - \lambda_{\text{min}}\varepsilon$, with the inequality resulting from matching up the m copies of $\lambda_{\text{min}} \max_e \mathbb{R}_e$

with the terms in the sum and noticing that each pair has a non-negative value (since $\mathbb{R}_i - \max_e \mathbb{R}_e$ is non-positive and $\lambda_{\min}$ is negative), and at least one pair has the value $-\lambda_{\min}\varepsilon$. Thus sending $\lambda \to -\infty$ sends this lower bound on $\mathbb{R}_{\text{MM-REx}}$ to ∞ and hence $\mathbb{R}_{\text{MM-REx}} \to \infty$ as well. □

Proposition 5. *The V-REx risk of predictor f_θ, $\mathbb{R}_{\text{V-REx}}(\theta) \to \infty$ as $\beta \to \infty$ unless $\mathbb{R}^d = \mathbb{R}^e$ for all training domains d, e.*

Proof. Again, let $\varepsilon > 0$ be the largest difference in training risks, and let μ be the mean of the training risks. Then there must exist an e such that $|\mathbb{R}_e - \mu| \geq \varepsilon/2$. And thus $Var_i(\mathbb{R}_i(\theta)) = \sum_i (\mathbb{R}_i - \mu)^2 \geq (\varepsilon/2)^2$, since all other terms in the sum are non-negative. Since $\varepsilon > 0$ by assumption, the penalty term is positive and thus $\mathbb{R}_{\text{V-REx}}(\theta) \doteq \sum_i \mathbb{R}_i(\theta) + \beta Var_i(\mathbb{R}_i(\theta))$ goes to infinity as $\beta \to \infty$. □

11.1.8.2 Connecting MM-REx to V-REx

11.1.8.2.1 Closed form solutions to risk interpolation and minimax-REx Here, we show that risk interpolation is equivalent to the robust optimization objective of Eqn. 5.5. Without loss of generality, let $\mathbb{R}_1$ be the largest risk, so $\mathbb{R}_e \leq \mathbb{R}_1$, for all e. Thus we can express $\mathbb{R}_e = \mathbb{R}_1 - d_e$ for some non-negative d_e, with $d_1 = 0 \geq d_e$ for all e. And thus we can write the weighted sum of Eqn. 5.7 as:

$$\mathbb{R}_{\text{MM}}(\theta) \doteq \max_{\substack{\sum_e \lambda_e = 1 \\ \lambda_e \geq \lambda_{\min}}} \sum_{e=1}^{m} \lambda_e \mathbb{R}_e(\theta) \tag{11.4}$$

$$= \max_{\substack{\sum_e \lambda_e = 1 \\ \lambda_e \geq \lambda_{\min}}} \sum_{e=1}^{m} \lambda_e (\mathbb{R}_1(\theta) - d_e) \tag{11.5}$$

$$= \mathbb{R}_1(\theta) + \max_{\substack{\sum_e \lambda_e = 2 \\ \lambda_e \geq \lambda_{\min}}} \sum_{e=1}^{m} -\lambda_e (d_e) \tag{11.6}$$

$$\tag{11.7}$$

Now, since d_e are non-negative, $-d_e$ is non-positive, and the maximal value of this sum is achieved when $\lambda_e = \lambda_{\min}$ for all $e \geq 2$, which also implies that $\lambda_1 = 1 - (m-1)\lambda_{\min}$.

This yields the closed form solution provided in Eqn. 5.7. The special case of Risk Interpolation, where $\lambda_{\min} = 0$, yields Eqn. 5.5.

11.1.8.2.2 Minimax-REx and Mean absolute error REx In the case of only two training risks, MM-REx is equivalent to using a penalty on the mean absolute error (MAE) between training risks. However, penalizing the pairwise absolute errors is not equivalent when there are $m > 2$ training risks, as we show below. Without loss of generality, assume that $\mathbb{R}_1 < \mathbb{R}_2 < ... < \mathbb{R}_m$. Then (1/2 of) the $\mathbb{R}_{\text{MAE}}$ penalty term is:

$$\sum_i \sum_{j \leq i} (\mathbb{R}_i - \mathbb{R}_j) = m\mathbb{R}_m - \sum_{j \leq m} \mathbb{R}_j + (m-1)\mathbb{R}_{m-1} - \sum_{j \leq m-1} \mathbb{R}_j \dots \tag{11.8}$$

$$= \sum_j j\mathbb{R}_j - \sum_j \sum_{i \leq j} \mathbb{R}_i \tag{11.9}$$

$$= \sum_j j\mathbb{R}_j - \sum_j (m-j+1)\mathbb{R}_j \tag{11.10}$$

$$= \sum_j (2j-m-1)\mathbb{R}_j \tag{11.11}$$

For $m = 2$, we have $1/2\mathbb{R}_{\text{MAE}} = (2*1-2-1)\mathbb{R}_1 + (2*2-2-1)\mathbb{R}_2 = \mathbb{R}_2 - \mathbb{R}_1$. Now, adding this penalty term with some coefficient β_{MAE} to the ERM term yields:

$$\mathbb{R}_{\text{MAE}} \doteq \mathbb{R}_1 + \mathbb{R}_2 + \beta_{\text{MAE}}(\mathbb{R}_2 - \mathbb{R}_1) = (1-\beta_{\text{MAE}})\mathbb{R}_1 + (1+\beta_{\text{MAE}})\mathbb{R}_2 \tag{11.12}$$

$$\tag{11.13}$$

We wish to show that this is equal to $\mathbb{R}_{\text{MM}}$ for an appropriate choice of learning rate γ_{MAE} and hyperparameter β_{MAE}. Still assuming that $\mathbb{R}_1 < \mathbb{R}_2$, we have that:

$$\mathbb{R}_{\text{MM}} \doteq (1-\lambda_{\min})\mathbb{R}_2 + \lambda_{\min}\mathbb{R}_1 \tag{11.14}$$

Choosing $\gamma_{\text{MAE}} = 1/2\gamma_{\text{MM}}$ is equivalent to multiplying $\mathbb{R}_{\text{MM}}$ by 2, yielding:

$$2\,\mathbb{R}_{\text{MM}} \doteq 2(1-\lambda_{\min})\,\mathbb{R}_2 + 2\lambda_{\min}\,\mathbb{R}_1 \tag{11.15}$$

Now, in order for $\mathbb{R}_{\text{MAE}} = 2\,\mathbb{R}_{\text{MM}}$, we need that:

$$2 - 2\lambda_{\min} = 1 + \beta_{\text{MAE}} \tag{11.16}$$

$$2\lambda_{\min} = 1 - \beta_{\text{MAE}} \tag{11.17}$$

$$\tag{11.18}$$

And this holds whenever $\beta_{\text{MAE}} = 1 - 2\lambda_{\min}$. When $m > 2$, however, these are not equivalent, since R_{MM} puts equal weight on all but the highest risk, whereas $\mathbb{R}_{\text{MAE}}$ assigns a different weight to each risk.

11.1.8.2.3 Penalizing pairwise mean squared error (MSE) yields V-REx The V-REx penalty (Eqn. 5.8) is equivalent to the average pairwise mean squared error between all training risks (up to a constant factor of 2). Recall that $\mathbb{R}_i$ denotes the risk on domain i. We have:

$$\frac{1}{2n^2}\sum_i\sum_j \left(\mathbb{R}_i - \mathbb{R}_j\right)^2 = \frac{1}{2n^2}\sum_i\sum_j \left(\mathbb{R}_i^2 + \mathbb{R}_j^2 - 2\,\mathbb{R}_i\,\mathbb{R}_j\right) \tag{11.19}$$

$$= \frac{1}{2n}\sum_i \mathbb{R}_i^2 + \frac{1}{2n}\sum_j \mathbb{R}_j^2 - \frac{1}{n^2}\sum_i\sum_j \mathbb{R}_i\,\mathbb{R}_j \tag{11.20}$$

$$= \frac{1}{n}\sum_i \mathbb{R}_i^2 - \left(\frac{1}{n}\sum_i \mathbb{R}_i\right)^2 \tag{11.21}$$

$$= \text{Var}(\mathbb{R})\,. \tag{11.22}$$

11.1.9 Further results and details for experiments mentioned in main text

11.1.9.1 CMNIST with covariate shift

Here we present the following additional results:

1. Figure 1 of the main text with additional results using MM-REx, see 11.5. These results used the "default" parameters from the code of Arjovsky et al. [8].

2. A plot with results on these same tasks after performing a random search over hyperparameter values similar to that performed by Arjovsky et al. [8].

3. A plot with the percentage of the randomly sampled hyperparameter combinations that have satisfactory ($> 50\%$) accuracy, which we count as "success" since this is better than random chance performance.

These results show that REx is able to handle greater covariate shift than IRM, given appropriate hyperparameters. Furthermore, when appropriately tuned, REx can outperform IRM in situations with covariate shift. The lower success rate of REx for high values of p is because it produces degenerate results (where training accuracy is less than test accuracy) more often.

The hyperparameter search consisted of a uniformly random search of 340 samples over the following intervals of the hyperparameters:

1. HiddenDim = [2**7, 2**12]

2. L2RegularizerWeight = [10**-2, 10**-4]

3. Lr = [10**-2.8, 10**-4.3]

4. PenaltyAnnealIters = [50, 250]

5. PenaltyWeight = [10**2, 10**6]

6. Steps = [201, 601]

11.1.9.2 SEMs from "Invariant Risk Minimization"

Here we present experiments on the (linear) structural equation model (SEM) tasks introduced by Arjovsky et al. [8]. Arjovsky et al. [8] construct several varieties of SEM where the task is to predict targets Y from inputs X_1, X_2, where X_1 are (non-anti-causal)

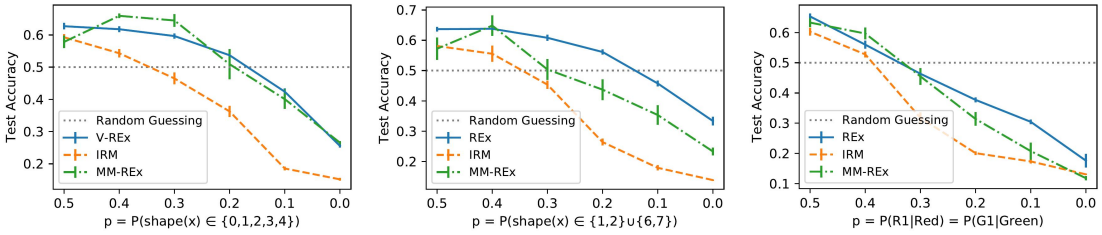


Figure 11.5: This is Figure 5.3.4 of main text with additional results using MM-REx. For each covariate shift variant (class imbalance, digit imbalance, and color imbalance from left to right as described in "CMNIST with covariate shift" subsubsection of Section 4.1 in main text) of CMNIST, the standard error (the vertical bars in plots) is higher for MM-REx than for V-REx.

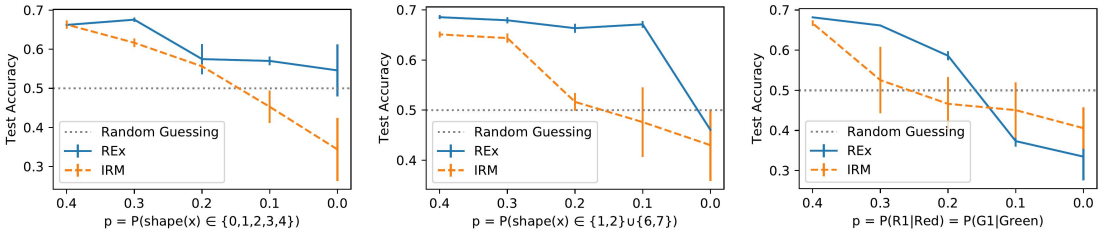


Figure 11.6: This is Figure 5.3.4 of main text (class imbalance, digit imbalance, and color imbalance from left to right as described in "CMNIST with covariate shift" subsubsection of Section 4.1 in main text), but with hyperparameters of REx and IRM each tuned to perform as well as possible for each value of p for each covariate shift type.

causes of Y, and X_2 are (anti-causal) effects of Y. We refer the reader to Section 5.1 and Figure 3 of Arjovsky et al. [8] for more details. We use the same experimental settings as Arjovsky et al. [8] (except we only run 7 trials), and report results in Table 11.II.

These experiments include several variants of a simple SEM, given by:

$$X_1 = N_1$$
$$Y = W_{1\to Y} X_1 + N_Y$$
$$X_2 = W_{Y\to 2} Y + N_2$$

Where N_1, N_Y, N_2 are all sampled i.i.d. from normal distributions. The variance of these distributions may vary across domains.

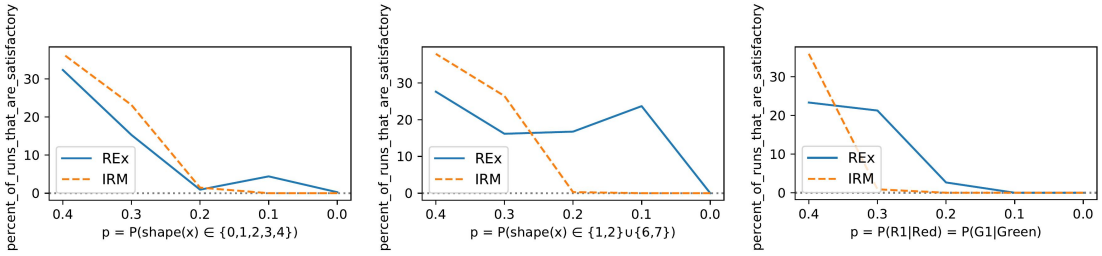


Figure 11.7: This also corresponds to class imbalance, digit imbalance, and color imbalance from left to right as described in "CMNIST with covariate shift" subsubsection of Section 4.1 in main text; but now the y-axis refers to what percentage of the randomly sampled hyperparameter combinations we deemed to to be satisfactory. We define satisfactory as simultaneously being better than random guessing and having train accuracy greater than test accuracy. For p less than .5, a larger percentage of hyperparameter combinations are often satisfactory for REx than for IRM; for p greater than .5, a larger percentage of hyperparameter combinations are often satisfactory for IRM than for REx because train accuracy is greater than test accuracy for more hyperparameter combinations for IRM. We stipulate that train accuracy must be greater than test accuracy because test accuracy being greater than train accuracy usually means the model has learned a degenerate prediction rule such as "not color".

While REx achieves good performance in the **domain-homoskedastic** case, it performs poorly in the **domain-heteroskedastic** case, where the amount of intrinsic noise, σ_y^2 in the target changes across domains.[2] Intuitively, this is because the irreducible error varies across domains in these tasks, meaning that the risk will be larger on some domains than others, even if the model's predictions match the expectation $\mathbb{E}(Y|Pa(Y))$. We tried using a "baseline" (see Eqn. 5.5) of $r_e = Var(Y_e)$ [212] to account for the different noise levels in Y, but this did not work.

We include a mathematical analysis of the simple SEM given above in order to better understand why REx succeeds in the domain-homoskedastic, but not the domain-heteroskedastic case. Assuming that Y, X_1, X_2 are scalars, this SEM becomes

$$
\begin{aligned}
X_1 &= N_1 \\
Y &= w_{1\to y} N_1 + N_Y \\
X_2 &= w_{y\to 2} w_{1\to y} N_1 + w_{y\to 2} N_Y + N_2
\end{aligned}
$$

[2] See Footnote 6.

We consider learning a model $\hat{Y} = \alpha X_1 + \beta X_2$. Then the residual is:

$$\hat{Y} - Y = (\alpha + w_{1\to y}(\beta w_{y\to 2} - 1))N_1 + (\beta w_{y\to 2} - 1)N_Y + \beta N_2$$

Since all random variables have zero mean, the MSE loss is the variance of the residual. Using the fact that the noise N_1, N_Y, N_2 are independent, this equals:

$$\mathbb{E}[(\hat{Y} - Y)^2] = (\alpha + w_{1\to y}(\beta w_{y\to 2} - 1))^2\sigma_1^2 + (\beta w_{y\to 2} - 1)^2\sigma_Y^2 + \beta^2\sigma_2^2$$

Thus when (only) σ_2 changes, the only way to keep the loss unchanged is to set the coefficient in front of σ_2 to 0, meaning $\beta = 0$. By minimizing the loss, we then recover $\alpha = w_{1\to y}$; i.e. in the domain-homoskedastic setting, the loss equality constraint of REx yields the causal model. On the other hand, if (only) σ_Y changes, then REx enforces $\beta = 1/w_{y\to 2}$, which then induces $\alpha = 0$, recovering the *anti*causal model.

While REx (like ICP [233]) assumes the mechanism for Y is fixed across domains (meaning $P(Y|Pa(Y))$ is independent of the domain, e), IRM makes the somewhat weaker assumption that $\mathbb{E}(Y|Pa(Y))$ is independent of domain. While it is plausible that an appropriately designed variant of REx could work under this weaker assumption, we believe forbidding interventions on Y is not overly restrictive, and such an extension for future work.

11.1.9.3 Reinforcement Learning Experiments

Here we provide details and further results on the experiments in Section 5.4.1. We take tasks from the Deepmind Control Suite [278] and modify the original state, $\mathbf{s}$, to produce observation, $\mathbf{o} = (\mathbf{s} + \varepsilon, \eta\mathbf{s}')$ including noise ε and spurious features $\eta\mathbf{s}'$, where $\mathbf{s}'$ contains 1 or 2 dimensions of $\mathbf{s}$. The scaling factor takes values $\eta = 1/2/3$ for the two training and test domains, respectively. The agent takes $\mathbf{o}$ as input and learns a representation using Soft Actor-Critic [128] and an auxiliary reward predictor, which is trained to predict the next 3 rewards conditioned on the next 3 actions. Since the spurious features are copied from the state before the noise is added, they are more informative for

	FOU(c)	FOU(nc)	FOS(c)	FOS(nc)
IRM	0.001±0.000	0.001±0.000	0.001±0.000	0.000±0.000
REx, $r_e = 0$	0.001±0.000	0.008±0.002	0.007±0.002	0.000±0.000
REx, $r_e = \mathbb{V}(Y_e)$	0.816±0.149	1.417±0.442	0.919±0.091	0.000±0.000

	POU(c)	POU(nc)	POS(c)	POS(nc)
IRM	0.004±0.001	0.006±0.003	0.002±0.000	0.000±0.000
REx, $r_e = 0$	0.004±0.001	0.004±0.001	0.002±0.000	0.000±0.000
REx, $r_e = \mathbb{V}(Y_e)$	0.915±0.055	1.113±0.085	0.937±0.090	0.000±0.000

	FEU(c)	FEU(nc)	FES(c)	FES(nc)
IRM	0.0053±0.0015	0.1025±0.0173	0.0393±0.0054	0.0000±0.0000
REx, $r_e = 0$	0.0390±0.0089	19.1518±3.3012	7.7646±1.1865	0.0000±0.0000
REx, $r_e = \mathbb{V}(Y_e)$	0.7713±0.1402	1.0358±0.1214	0.8603±0.0233	0.0000±0.0000

	PEU(c)	PEU(nc)	PES(c)	PES(nc)
IRM	0.0102±0.0029	0.0991±0.0216	0.0510±0.0049	0.0000±0.0000
REx, $r_e = 0$	0.0784±0.0211	46.7235±11.7409	8.3640±2.6108	0.0000±0.0000
REx, $r_e = \mathbb{V}(Y_e)$	1.0597±0.0829	0.9946±0.0487	1.0252±0.0819	0.0000±0.0000

Table 11.II: Average mean-squared error between true and estimated weights on causal (X_1) and non-causal (X_2) variables. **Top 2:** When the level of noise in the anti-causal features varies across domains, REx performs well (FOU, FOS, POU, POS). **Bottom 2:** When the level of noise in the targets varies instead, REx performs poorly (FEU, FES, PEU, PES). Using the baselines $r_e = \mathbb{V}(Y)$ does not solve the problem, and indeed, hurts performance on the homoskedastic domains.

the reward prediction task, but they do not have an invariant relationship with the reward because of the domain-dependent η.

The hyperparameters used for training Soft Actor-Critic can be found in Table 11.III. We used `cartpole_swingup` as a development task to tune the hyperparameters of penalty weight (chosen from $[0.01, 0.1, 1, 10]$) and number of iterations before the penalty is turned up (chosen from $[5000, 10000, 20000]$), both for REx and IRM. The plots with the hyperparameter sweep are in Figure 11.8.

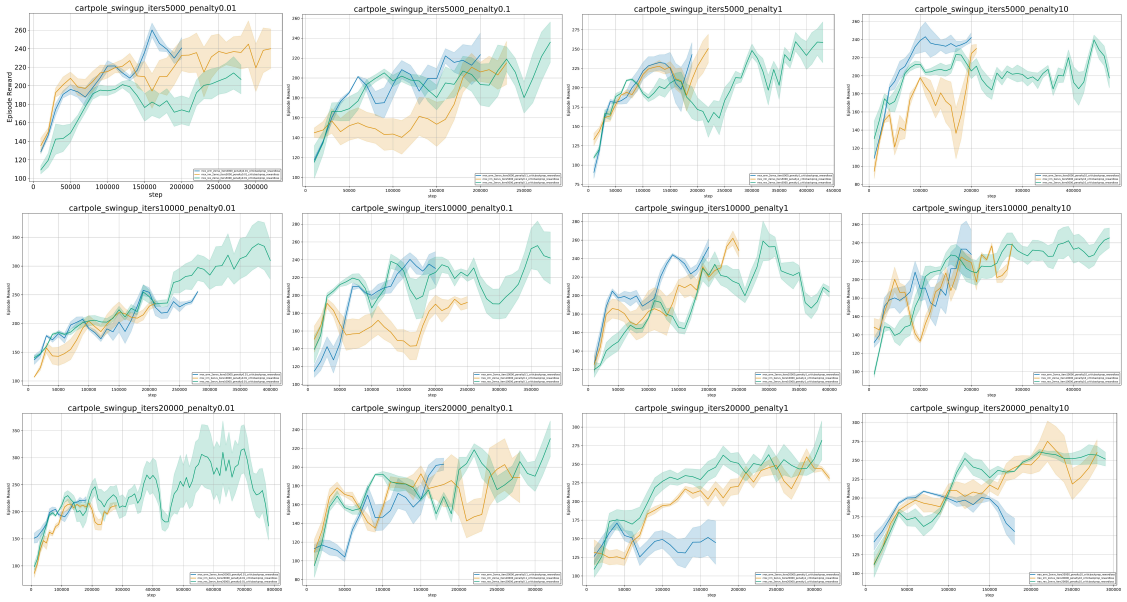


Figure 11.8: Hyperparameter sweep for IRM and REx on `cartpole_swingup`. Green, blue, and orange curves correspond to REx, ERM, and IRM, respectively. The subfigure titles state the penalty strength ("penalty") and after how many iterations the penalty strength was increased ("iters"). We chose a penalty factor of 1 and 10k iterations.

11.1.10 Experiments not mentioned in main text

We include several other experiments which do not contribute directly to the core message of our paper. Here is a summary of the take-aways from these experiments:

1. Our experiments in the CMNIST domain suggest that the IRM/V-REx penalty terms should be amplified exactly when the model starts overfitting training distributions.

2. Our financial indicators experiments suggest that IRM and REx often perform remarkably similarly in practice.

11.1.10.1 A possible approach to scheduling IRM/REx penalties

We've found that REx and IRM are quite sensitive to the choice of hyperparameters. In particular, hyperparameters controlling the scheduling of the IRM/V-REx penalty terms are of critical importance. For the best performance, the penalty should be increased the relative weight of the penalty term after approximately 100 epochs of training (using a

Parameter name	Value
Replay buffer capacity	1000000
Batch size	1024
Discount γ	0.99
Optimizer	Adam
Critic learning rate	10^{-5}
Critic target update frequency	2
Critic Q-function soft-update rate τ_{Q}	0.005
Critic encoder soft-update rate τ_{enc}	0.005
Actor learning rate	10^{-5}
Actor update frequency	2
Actor log stddev bounds	$[-5, 2]$
Encoder learning rate	10^{-5}
Decoder learning rate	10^{-5}
Decoder weight decay	10^{-7}
L1 regularization weight	10^{-5}
Temperature learning rate	10^{-4}
Temperature Adam's β_1	0.9
Init temperature	0.1

Table 11.III: A complete overview of hyperparameters used for reinforcement learning experiments.

so-called "waterfall" schedule [72]). See Figure 11.9(b) for a comparison. We also tried an exponential decay schedule instead of the waterfall and found the results (not reported) were significantly worse, although still above 50% accuracy.

Given the methodological constraints of out-of-distribution generalization mentioned in [123], this could be a significant practical issue for applying these algorithms. We aim to address this limitation by providing a guideline for when to increase the penalty weight, based only on the training domains. We hypothesize that successful learning of causal features using REx or IRM should proceed in two stages:

1. In the first stage, predictive features are learned.

2. In the second stage, causal features are selected and/or predictive features are fine-tuned for stability.

This viewpoint suggests that we could use overfitting on the *training* tasks as an indicator

for when to apply (or increase) the IRM or REx penalty.

The experiments presented in this section provide *observational* evidence consistent with this hypothesis. However, since the hypothesis was developed by observing patterns in the CMNIST training runs, it requires further experimental validation on a different task, which we leave for future work.

11.1.10.1.1 Results and Interpretation In Figure 11.9, we demonstrate that the optimal point to apply the waterfall in the CMNIST task is after predictive features have been learned, but before the model starts to memorize training examples. Before predictive features are available, the penalty terms push the model to learn a constant predictor, impeding further learning. And after the model starts to memorize, it become difficult to distinguish anti-causal and causal features. This second effect is because neural networks often have the capacity to memorize all training examples given sufficient training time, achieving and near-0 loss [316]. In the limits of this memorization regime, the differences between losses become small, and gradients of the loss typically do as well, and so the REx and IRMv1 penalties no longer provide a strong or meaningful training signal, see Figure 11.10.

11.1.10.2 Domain Generalization: VLCS and PACS

Here we provide earlier experiments on the VLCS and PACS dataset. We removed these experiments from the main text of our paper in favor of the more complete DomainBed results.

To test whether REx provides a benefit on more realistic domain generalization tasks, we compared REx, IRM and ERM performance on the VLCS [286] and PACS [192] image datasets. Both datasets are commonly-used for multi-source domain generalization. The task is to train on three domains and generalize to a fourth one at test time.

Since every domain in PACS is used as a test set when training on the other three domains, it is not possible to perform a methodologically sound evaluation on PACS after examining results on *any* of the data. Thus to avoid performing any tuning on test distributions, we use VLCS to tune hyperparameters and then apply these exact same

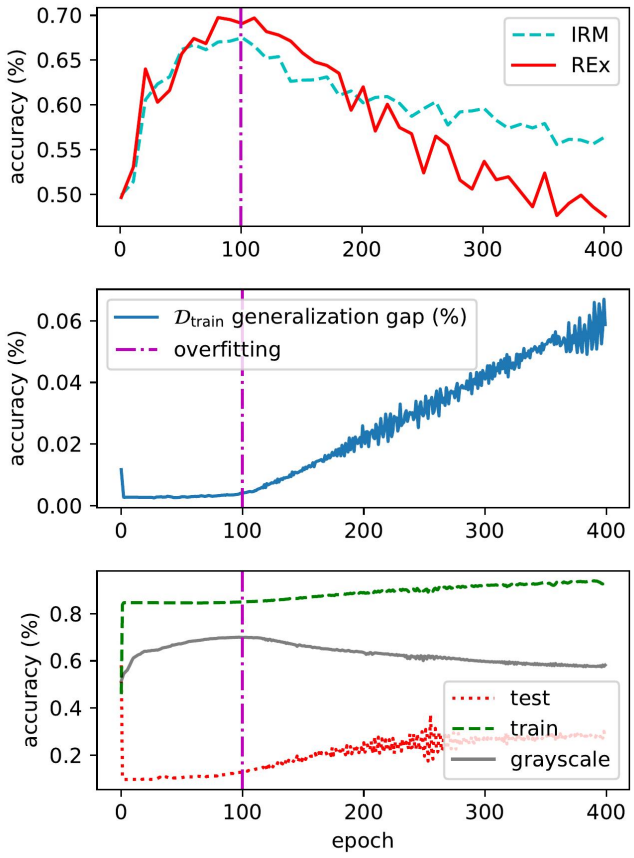


Figure 11.9: Stability penalties should be applied around when traditional overfitting begins, to ensure that the model has learned predictive features, and that penalties still give meaningful training signals. **Top:** Test accuracy as a function of epoch at which penalty term weight is increased (learning rate is simultaneously decreased proportionally). Choosing this hyperparameter correctly is essential for good performance. **Middle:** Generalization gap on a validation set with 85% correlation between color and label (the same as the average training correlation). The best test accuracy is achieved by increasing the penalty when the generalization gap begins to increase. The increase clearly indicates memorization because color and shape are only 85%/75% correlated with the label, and so cannot be used to make predictions with higher than 85% accuracy. **Bottom:** Accuracy on training/test sets, as well as an auxilliary grayscale set. Training/test performance reach 85%/15% after a few epochs of training, but grayscale performance improves, showing that meaningful features are still being learned.

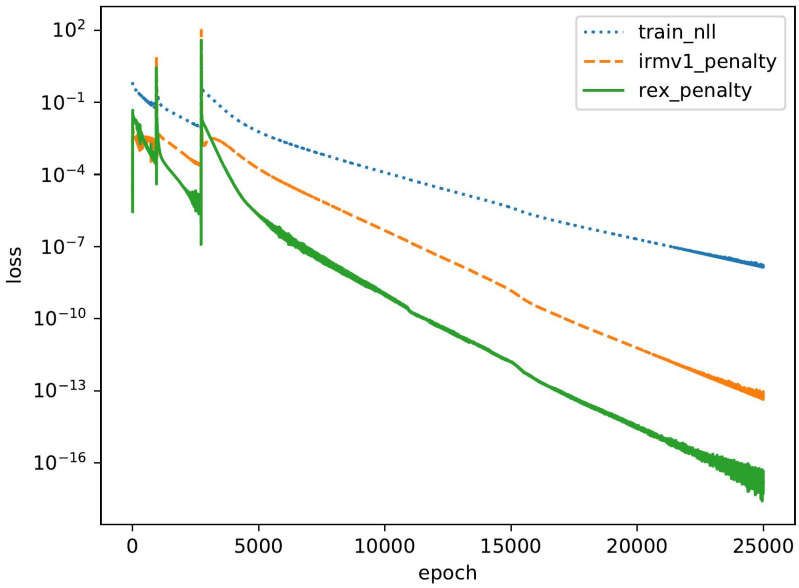


Figure 11.10: Given sufficient training time, empirical risk minimization (ERM) minimizes both REx and IRMv1 penalty terms on Colored MNIST (*without* including either term in the loss function). This is because the model (a deep network) has sufficient capacity to fit the training sets almost perfectly. This prevents these penalties from having the intended effect, once the model has started to overfit. The y-axis is in log-scale.

settings to PACS and report the final average over 10 runs on each domain.

We use the same architecture, training procedure and data augmentation strategy as the (formerly) state-of-the-art Jigsaw Puzzle approach [46] (except with IRM or V-REx intead of JigSaw as auxilliary loss) for all three methods. As runs are very noisy, we ran each experiment 10 times, and report average test accuracies extracted at the time of the highest validation accuracy on each run. Results on PACS are in Table 11.V. On PACS we found that REx outperforms IRM and IRM outperforms ERM on average, while all are worse than the state-of-the-art Jigsaw method.

We use all hyperparameters from the original Jigsaw codebase.[3] We use Imagenet pre-trained AlexNet features and chose batch-size, learning rate, as well as penalty weights based on performance on the VLCS dataset where test performance on the holdout do-

[3] `https://github.com/fmcarlucci/JigenDG`

main was used for the set of parameters producing the highest validation accuracy. The best performing parameters on VLCS were then applied to the PACS dataset without further changes. We searched over batch-sizes in $\{128, 384\}$, over penalty strengths in $\{0.0001, 0.001, 0.01, 0.1, 1, 10\}$, learning rates in $\{0.001, 0.01\}$ and used average performance over all 4 VLCS domains to pick the best performing hyperparameters. Table 11.IV shows results on VLCS with the best performing hyperparameters.

The final parameters for all methods on PACS were a batch size of 384 with 30 epochs of training with Adam, using a learning rate of 0.001, and multiplying it by 0.1 after 24 epochs (this step schedule was taken from the Jigsaw repo).The penalty weight chosen for Jigsaw was 0.9; for IRM and REx it was 0.1.We used the same data-augmentation pipeline as the original Jigsaw code for ERM, IRM, Jigsaw and REx to allow for a fair comparison.

VLCS	**CALTECH**	**SUN**	**PASCAL**	**LABELME**	**Average**
REx (ours)	~~**96.72**~~	~~63.68~~	~~**72.41**~~	~~60.40~~	~~**73.30**~~
IRM	~~95.99~~	~~62.85~~	~~71.71~~	~~59.61~~	~~72.54~~
ERM	~~94.76~~	~~61.92~~	~~69.03~~	~~**60.55**~~	~~71.56~~
Jigsaw (SOTA)	~~96.46~~	~~**63.84**~~	~~70.49~~	~~60.06~~	~~72.71~~

Table 11.IV: Accuracy (percent) of different methods on the VLCS task. Results are test accuracy at the time of the highest validation accuracy, averaged over 10 runs. On VLCS REx outperforms all other methods. Numbers are shown in strike-through because we selected our hyperparameters based on highest test set performance; the goal of this experiment was to find suitable hyperparameters for the PACS experiment.

PACS	**Art Painting**	**Cartoon**	**Sketch**	**Photo**	**Average**
REx (ours)	66.27±0.46	68.8±0.28	59.57±0.78	89.60±0.12	71.07
IRM	66.46±0.31	68.60±0.40	58.66±0.73	89.94±0.13	70.91
ERM	66.01±0.22	68.62±0.36	58.38±0.60	89.40±0.18	70.60
Jigsaw (SOTA)	66.96±0.39	66.67±0.41	61.27±0.73	89.54±0.19	71.11

Table 11.V: Accuracy (percent) of different methods on the PACS task. Results are test accuracy at the time of the highest validation accuracy, averaged over 10 runs. REx outperforms ERM on average, and performs similar to IRM and Jigsaw (the state-of-the-art).

11.1.10.3 Financial indicators

We find that IRM and REx seem to perform similarly across different splits of the data in a prediction task using financial data. The dataset is split into five years, 2014–18, containing 37 publicly reported financial indicators of several thousand publicly listed companies each. The task is to predict if a company's value will increase or decrease in the following year (see Appendix for dataset details.) We consider each year a different domain, and create 20 different tasks by selecting all possible combinations of domains where three domains represent the training sets, one domain the validation set, and another one the test set. We train an MLP using the validation set to determine an early stopping point, with $\beta = 10^4$. The per-task results summarized in fig. 11.11 indicate substantial differences between ERM and IRM, and ERM and REx. The predictions produced by IRM and REx, however, only differ insignificantly, highlighting the similarity of IRM and REx. While performance on specific tasks differs significantly between ERM and IRM/REx, performance averaged over tasks is not significantly different.

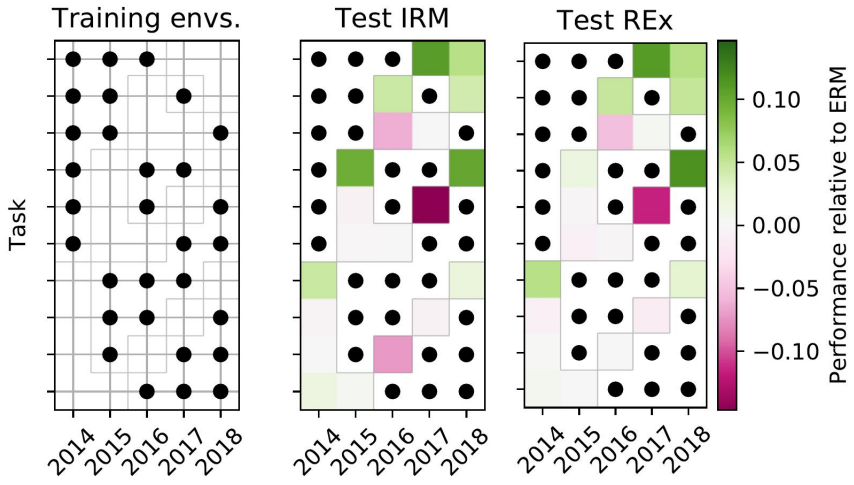


Figure 11.11: Financial indicators tasks. The left panel indicates the set of training domains; the middle and right panels show the test accuracy on the respective domains relative to ERM (a black dot corresponds to a training domain; a colored patch indicates the test accuracy on the respective domain.)

11.1.10.3.1 Experiment Details We use `v1` of the dataset published on [4] and prepare the data as described in.[5] We further remove all the variables that are not shared across all 5 years, leaving us with 37 features, and whiten the data through centering and normalizing by the standard deviation.

On each subtask, we train an MLP with two hidden layers of size 128 with tanh activations and dropout (p=0.5) after each layer. We optimize the binary cross-entropy loss using Adam (learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$), and an L2 penalty (weight 0.001). In the IRM/REx experiments, the respective penalty is added to the loss ($\beta = 1$) and the original loss is scaled by a factor 10^{-4} after 1000 iterations. Experiments are run for a maximum of 9000 training iterations with early stopping based on the validation performance. All results are averaged over 3 trials. The overall performance of the different models, averaged over all tasks, is summarized in Tab. 11.VI. The difference in average performance between ERM, IRM, and REx is not statistically significant, as the error bars are very large.

	Overall accuracy	Min acc.	Max acc.
ERM	54.6 ± 4.6	47.6	66.2
IRM	55.3 ± 5.9	45.9	67.5
REx	$\mathbf{55.5 \pm 6.0}$	47.2	**68.0**

Table 11.VI: Test accuracy of models trained on the financial domain dataset, averaged over all 20 tasks, as well as min./max. accuracy across the tasks.

11.1.11 Overview of other topics related to OOD generalization

Domain adaptation [17] shares the goal of generalizing to new distributions at test time, but allows some access to the test distribution. A common approach is to make different domains have a similar distribution of features [229]. A popular deep learning method for doing so is Adversarial Domain Adaptation (ADA) [100, 193, 201, 290],

[4] `https://www.kaggle.com/cnic92/200-financial-indicators-of-us-stocks-20142018`

[5] `https://www.kaggle.com/cnic92/explore-and-clean-financial-indicators-dataset`

which seeks a "invariant representation" of the inputs, i.e. one whose distribution is domain-independent. Recent works have identified fundamental shortcomings with this approach, however [8, 156, 311, 319].

Complementary to the goal of domain generalization is **out-of-distribution detection** [139, 140], where the goal is to recognize examples as belonging to a new domain. Three common deep learning techniques that can improve OOD generalization are **adversarial training** [115, 137], **self-supervised learning** [2, 142, 145, 296] and **data augmentation** [46, 64, 141, 174, 260, 318]. These methods can also been combined effectively in various ways [11, 119, 283]. Data augmentation and self-supervised learning methods typically use prior knowledge such as 2D image structure. Several recent works also use prior knowledge to design augmentation strategies for invariance to superficial features that may be spuriously correlated with labels in object recognition tasks [119, 135, 153, 301]. In contrast, REx can discover which features have invariant relationships with the label without such prior knowledge.

11.2 Appendix for "Bayesian Hypernetworks"

11.2.1 Additional results

11.2.1.1 Learning correlated weights

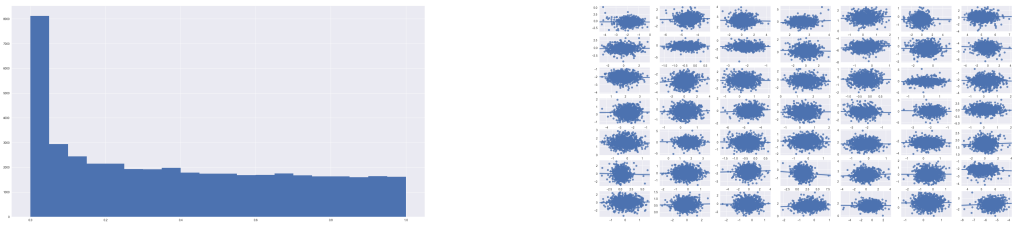


Figure 11.12: Histogram of Pearson correlation coefficient p-values (left) and a scatter matrix (right) of samples from a hypernet approximate posterior. We see that the hypernet posterior includes correlations between different parameters. Many of the p-values of the Pearson correlation test are below .05.

11.2.1.2 Unseen mode detection

We replicate the experiments of anomaly detection with unseen classes of MNIST.

Table 11.VII: Anomaly detection on MNIST with unseen classes. The first column indicates the missing class label in the training set. Top-most block: ROC score; middle: positive precision-recall; bottom: negative precision-recall.

	Variation ratio				Mean std			BALD		
	MLP	dropout	BHN 4	BHN 8	dropout	BHN 4	BHN 8	dropout	BHN 4	BHN 8
0	95.52	97.44	96.62	96.45	**97.90**	96.53	96.77	97.89	96.59	96.55
1	96.70	94.60	96.62	96.46	94.01	96.62	96.25	93.92	**96.92**	96.19
2	92.83	95.77	92.99	93.47	96.02	93.03	93.57	**96.08**	93.59	94.26
3	93.03	93.11	95.03	**95.34**	93.65	94.86	94.77	93.65	94.87	94.96
4	89.08	88.96	75.73	81.19	**89.45**	75.73	81.31	89.34	74.31	84.34
5	88.53	94.66	93.20	87.95	95.37	93.08	88.31	**95.45**	92.61	85.77
6	95.40	96.33	93.67	94.69	**96.99**	93.80	94.80	96.96	93.27	94.50
7	92.46	96.61	95.08	93.70	**97.08**	94.68	92.82	97.06	94.88	92.89
8	96.35	**98.05**	95.86	96.85	97.67	95.74	96.98	97.23	95.48	96.87
9	94.75	95.95	95.62	**96.54**	96.03	95.46	96.42	96.10	95.84	96.37
0	97.68	98.68	98.34	98.32	**98.87**	98.31	98.45	**98.87**	98.35	98.35
1	98.26	97.03	98.23	98.15	96.58	98.20	98.04	96.58	**98.35**	98.00
2	96.06	97.74	95.63	96.07	97.83	95.31	96.01	**97.87**	95.80	96.45
3	96.00	95.74	97.28	**97.68**	95.97	97.09	97.37	96.00	97.13	97.49
4	93.73	93.93	84.66	86.40	**94.10**	85.16	86.46	94.00	83.32	90.00
5	93.92	97.31	96.79	93.15	97.60	96.62	93.34	**97.61**	96.34	90.72
6	97.68	97.99	96.38	97.27	**98.29**	96.55	97.29	**98.29**	96.05	97.13
7	95.56	98.16	97.40	96.51	**98.36**	97.07	95.82	98.32	97.17	95.89
8	98.18	**99.03**	97.97	98.37	98.87	97.96	98.53	98.70	97.83	98.45
9	97.32	97.94	97.76	98.27	97.93	97.71	**98.31**	98.02	98.00	98.29
0	90.11	94.44	92.17	90.95	**96.08**	92.06	92.67	**96.08**	91.92	91.91
1	92.84	89.08	92.48	91.99	88.11	92.71	91.53	87.67	**93.11**	91.55
2	85.74	91.13	86.61	87.52	92.04	88.22	88.51	**92.16**	89.20	89.90
3	87.46	87.78	89.46	88.75	89.72	89.99	87.09	89.78	**90.22**	87.21
4	80.96	79.04	64.02	72.11	81.82	64.33	73.69	**81.89**	64.16	75.72
5	80.41	87.74	84.15	78.16	90.48	84.85	78.96	**90.81**	84.27	76.99
6	89.34	92.26	88.17	88.60	**94.21**	88.28	89.10	94.07	87.14	87.89
7	87.08	92.69	88.91	86.85	94.02	89.07	86.64	**94.33**	89.70	86.71
8	91.88	**95.82**	90.52	92.83	94.40	89.69	92.82	92.80	88.41	92.08
9	88.10	90.71	89.70	**91.67**	91.49	88.72	90.85	91.56	88.79	90.30

11.2.1.3 Stronger attack

Here we use 32 samples to estimate the gradient direction with respect to the input. A better estimate of gradient amounts to a stronger attack, so accuracy drops lower for

a given step size while an adversarial example can be more easily detected with a more informative uncertainty measure.

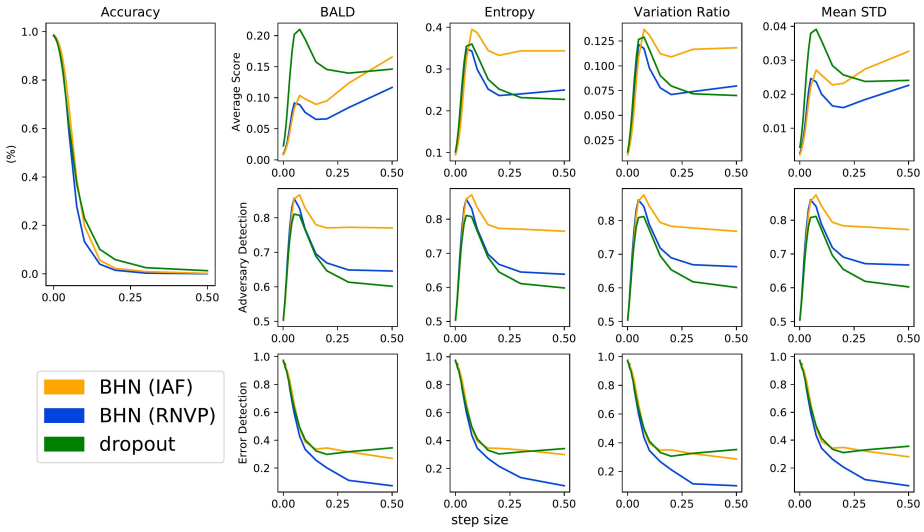


Figure 11.13: Adversary detection with 32-sample estimate of gradient.

11.2.2 Derivation of training objective

In this paper, we employ weight normalization in the primary network (7.7), treating (only) the scaling factors $\mathbf{g}$ as random variables. We choose an isotropic Gaussian prior for $\mathbf{g}$: $p(\mathbf{g}) = \mathcal{N}(\mathbf{g}; \mathbf{0}, \lambda \mathbf{I})$, which results in an L_2 weight-decay penalty on $\mathbf{g}$, or, equivalently, $\mathbf{w} = \mathbf{g}\frac{\mathbf{v}}{||\mathbf{v}||_2}$. Our objective and lower bound are then:

$$\log p(\mathscr{D}; \mathbf{v}, \mathbf{b}) = \log \int_{\mathbf{g}} p(\mathscr{D}|\mathbf{g}; \mathbf{v}, \mathbf{b}) p(\mathbf{g}) d\mathbf{g} \tag{11.23}$$

$$\geq \mathbb{E}_{q(\mathbf{g})}[\log p(\mathscr{D}|\mathbf{g}; \mathbf{v}, \mathbf{b}) + \log p(\mathbf{g}) - \log q(\mathbf{g})] \tag{11.24}$$

$$\geq \mathbb{E}_{\varepsilon \sim q_\varepsilon(\varepsilon), \mathbf{g}=h_\phi(\varepsilon)}[\log p(\mathscr{D}|\mathbf{g}; \mathbf{v}, \mathbf{b}) + \log p(\mathbf{g}) - \log q(\varepsilon) + \log \left| \det \frac{\partial h_\phi(\varepsilon)}{\partial \varepsilon} \right|] \tag{11.25}$$

where $\mathbf{v}$ and $\mathbf{b}$ are the direction and bias parameters of the primary net, and ϕ is the

parameters of the hypernetwork. We optimize this bound with respect to $\{\mathbf{v}, \mathbf{b}, \phi\}$ during training.