Generalization Through the Lens of Learning Dynamics



Clare Lyle

University College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Abstract

A machine learning (ML) system must learn not only to match the output of a target function on a training set, but also to generalize to novel situations in order to yield accurate predictions at deployment. In most practical applications, the user cannot exhaustively enumerate every possible input to the model; strong generalization performance is therefore crucial to the development of ML systems which are performant and reliable enough to be deployed in the real world. While generalization is well-understood theoretically in a number of hypothesis classes, the impressive generalization performance of deep neural networks has stymied theoreticians. In deep reinforcement learning (RL), our understanding of generalization is further complicated by the conflict between generalization and stability in widely-used RL algorithms. This thesis will provide insight into generalization by studying the learning dynamics of deep neural networks in both supervised and reinforcement learning tasks.

We begin with a study of generalization in supervised learning. We propose new PAC-Bayes generalization bounds for invariant models and for models trained with data augmentation. We go on to consider more general forms of inductive bias, connecting a notion of training speed with Bayesian model selection. This connection yields a family of marginal likelihood estimators which require only sampled losses from an iterative gradient descent trajectory, and analogous performance estimators for neural networks. We then turn our attention to reinforcement learning, laying out the learning dynamics framework for the RL setting which will be leveraged throughout the remainder of the thesis. We identify a new phenomenon which we term capacity loss, whereby neural networks lose their ability to adapt to new target functions over the course of training in deep RL problems, for which we propose a novel regularization approach. Follow-up analysis studying more subtle forms of capacity loss reveals that deep RL agents are prone to memorization due to the unstructured form of early prediction targets, and highlights a solution in the form of distillation. We conclude by calling back to a different notion of invariance to that which started this thesis, presenting a novel representation learning method which promotes invariance to spurious factors of variation in the environment.

Acknowledgements

First and foremost, this thesis would not have been possible without the invaluable guidance and support of my supervisors, Yarin Gal and Marta Kwiatkowska. Their advice, mentorship, and insight over the past almost four years has been vital to my development as a scientist. I am deeply grateful to both for being so incredibly generous with their time, particularly in the early years of the DPhil, and for giving me the freedom to set my own research direction and explore a broad range of topics. Thanks are also due to Marc Bellemare, Pablo Samuel Castro, and Prakash Panangaden for their mentorship prior to the start of my DPhil. I learned many invaluable lessons about how to do good research from their example, which have served me well throughout my DPhil.

I have had the good fortune to engage in a number of collaborations while at Oxford. Many thanks go to Amy Zhang, with whom I wrote the first published paper of my DPhil, for modelling how to organize and execute a research project. I am also indebted to Mark Rowland and Will Dabney, who have been a joy to work with on many of the papers that appear in this document. This thesis would not have been possible without valuable discussions and collaborations with Benjamin Bloem-Reddy, Mark van der Wilk, Benjie Wang, Angelos Filos, Natasha Jaques, Greg Farquhuar, Lisa Schut, Robin Ru, Aidan Gomez, Lorenz Kuhn, Jannick Kossen, Neil Band, Georg Ostrovski, Shagun Sodhani, and Andreas Kirsch.

Beyond direct collaborations, I also benefited immensely from being part of the broader machine learning community at Oxford. My perspective as a researcher was enriched by having a group of brilliant people with whom I could run over nascent ideas, debug a tricky experiment, and gripe about Reviewer 2. Thanks go in particular to Joost, Milad, Seb, Panos, Tim, Jan, Freddie, Pascal, Matt, Pascale, Rhiannon, Luca, Andrea, Emi, Michael, Sahra, Charline, and Jakob. Thanks as well go to the many people I interacted with at DeepMind, including Daniel, Dave, Remi, Diana, Anna, Bilal, Bernardo, and Mo.

Finally, I would be remiss to omit my family, whose unconditional support has provided the solid foundation on which I have been able to take risks and and explore as a researcher. Thanks go as well to the friends I've made in Oxford: Alex, Caitlin, Nayani, Colin, Matt, Rahul, Imi, Becca, and my teammates from OUBbC and UCBC. Thanks in particular to the individual who did not want to be named in this document for challenging me to address meaningful problems and for proofreading many chapters of this thesis.

Contents

List of Figures vi

List of Tables ix

1 Introduction 1
- 1.1 Learning to generalize . . . 1
- 1.2 Learning to act . . . 5
- 1.3 Understanding the learning process . . . 7
- 1.4 Thesis contributions and structure . . . 9

2 Background & literature review 16
- 2.1 Learning frameworks . . . 16
- 2.2 Generalization in supervised learning . . . 24
- 2.3 Generalization in reinforcement learning . . . 33

3 The role of invariance in generalization 44
- 3.1 Introduction . . . 44
- 3.2 Background . . . 46
- 3.3 Invariance and optimization dynamics . . . 50
- 3.4 Invariance and generalization bounds . . . 58
- 3.5 Conclusions . . . 64

4 Training speed and model selection 66
- 4.1 Introduction . . . 66
- 4.2 Background on Bayesian inference . . . 69
- 4.3 Marginal likelihood estimation with training statistics . . . 72
- 4.4 Training speed and interference . . . 80
- 4.5 Conclusions . . . 88

5 Dynamics of reinforcement learning 90
- 5.1 Introduction . . . 90
- 5.2 Mathematical framework . . . 92
- 5.3 Value function & representation dynamics . . . 94

5.4 Auxiliary tasks . . . 100
5.5 Experiments . . . 109
5.6 Conclusions . . . 113

6 Capacity loss **115**
6.1 Introduction . . . 115
6.2 Learning capacity in neural networks . . . 119
6.3 InFeR: mitigating capacity loss with feature regularization . . . 128
6.4 Conclusions . . . 133

7 Interference and generalization **135**
7.1 Introduction . . . 135
7.2 Learning dynamics and smoothness . . . 138
7.3 Generalization and interference in deep RL . . . 147
7.4 Post-training distillation and generalization . . . 153
7.5 Conclusions . . . 158

8 Generalization across environments **161**
8.1 Introduction . . . 161
8.2 Background on causality . . . 164
8.3 Problem setting . . . 166
8.4 State abstractions and causal feature sets . . . 172
8.5 State abstractions for rich observations . . . 179
8.6 Conclusions . . . 188

9 Conclusion **189**

Appendices

A The role of invariance in generalization **196**
A.1 Proofs . . . 196
A.2 Examples, counterexamples, tighter bounds . . . 202
A.3 Computation details for PAC-Bayes bounds . . . 203

B Training speed and model selection **205**
B.1 Linear model combination . . . 205
B.2 Proofs of theoretical results . . . 209
B.3 Experiments . . . 217

C Dynamics of reinforcement learning **221**
C.1 Auxiliary results . . . 221
C.2 Proofs . . . 224
C.3 Additional results from Table 5.1 . . . 229
C.4 Beyond diagonalisability assumptions . . . 232
C.5 Extensions beyond one-step temporal difference learning . . . 233
C.6 Bayes-optimality of RSBFs . . . 234
C.7 Further discussion of features and operator decompositions . . . 236
C.8 Learning dynamics for ensemble prediction . . . 238
C.9 Experimental details . . . 239

D Capacity loss **246**
D.1 Proofs . . . 246
D.2 Sequential supervised learning . . . 250
D.3 Atari evaluations . . . 254

E Interference and generalization **265**
E.1 Proofs . . . 265
E.2 Experiment details . . . 272
E.3 Additional numerical evaluations . . . 274
E.4 Additional empirical results . . . 277

F Generalization across environments **280**
F.1 Implementation details . . . 280

Bibliography **285**

List of Figures

1.1 A visualization of a fully connected multi-layer perceptron. 4
1.2 A figure visualizing the RL problem; featuring the environment (pictured on the left) and agent (right) dichotomy. 6

2.1 State abstractions allow an agent to treat similar states as though they were equivalent, potentially simplifying planning and generalization. 34
2.2 Visualization of multi-environment generalization framework with coinrun levels sampled from the ProcGen benchmark [Cobbe et al., 2020]. 39

3.1 A visualization of a fully connected multi-layer perceptron. 53
3.2 Measurements over the course of training a CNN using different DA and FA approaches. 54
3.3 Comparing the ranks of gradients computed on minibatches consisting of 32 independent random input samples, or 8 random samples from the training set augmented with all 90 degree rotations (resulting in 32 non-independent samples). 57

4.1 Visualization of the relationship between the Bayesian marginal likelihood and training speed. 67
4.2 Evaluation of our estimators. 79
4.3 Illustration of marginal likelihood estimation via our method on NTK models. 80
4.4 Effect of minibatch gradient steps on train, test, and minibatch loss over the course of training a variety of neural network architectures on CIFAR-10. 84
4.5 Linear combinations of DNNs on FashionMNIST. 85
4.6 Rankings of a randomly-selected subset of models from the NASBench-201 dataset given by training speed estimators. 87

5.1 An example of qualitatively different value function dynamics for a two-state MDP for 1-step temporal difference learning and Monte Carlo learning, with fixed point V^π in red. 91

5.2 Visualizing the Grassmann distance between two 2-dimensional subspaces. . . . 94
5.3 Study of feature dynamics. . . . 103
5.4 Transfer of EBFs, RSBFs, and RFs across the value-improvement path of a chain MDP, with and without the value function as an additional feature. . . . 110
5.5 Learning curves for DDQN, DDQN+RC, DDQN+Ensemble, DDQN+REM, and QR-DQN agents on several dense reward ALE environments. . 111
5.6 Sweep over number of auxiliary heads for RC, REM and Ensemble. 111

6.1 Deeper analysis on a sequential MNIST prediction setting showing that target-fitting capacity sees the greatest reduction in low capacity networks with ReLU units. . . . 121
6.2 Neural networks exhibit a decline in their ability to fit random network outputs over the course of training in two demonstrative Atari environments. . . . 123
6.3 Feature rank and performance over the course of training. . . . 125
6.4 Evaluation of the correlation between feature rank and performance. 127
6.5 Visualization and evaluation of InFeR. . . . 129
6.6 Deeper investigation into the effect of InFeR. . . . 132

7.1 Value function evolution and eigendecomposition on the MountainCar environment. . . . 139
7.2 Networks trained to fit high-frequency target functions exhibit pathological interpolation properties when later fine-tuned on a value function. . . . 143
7.3 Update matrices for a toy kernel regression problem. . . . 149
7.4 Evolution of interference in DQN agents trained on Atari. . . . 151
7.5 Update dimension of actor-critic methods in ProcGen. . . . 152
7.6 Performance and robustness to perturbations of different distillation approaches in games from the Atari suite. . . . 155
7.7 Effect of policy distillation on generalization in environments from the Procgen suite. . . . 156

8.1 Graphical model of a block MDP with emissions x from a state which can be decomposed into a causal component s and a spurious component η. . . . 169
8.2 Illustration of a causal state abstraction where the observation is a concatenation of variables from a causal graph. . . . 174
8.3 The presence of spurious uncorrelated variables in the state can still lead to poor generalization of linear function approximation methods. 179

8.4 Evaluation of our nonlinear MISA algorithm. 185
8.5 The Cheetah Run environment from Deepmind Control with different camera angles. The first two images are from the training environments and the last image is from evaluation environment. . . 186
8.6 Performance of nonlinear MISA on RL tasks. 187

List of Tables

3.1 Generalization performance for a permutation-invariant point cloud classification task (see text for details). 64

5.1 Summary of dynamics and limiting solutions under some common auxiliary tasks in the limit of infinitely-many prediction outputs. . . 100

1 Introduction

Contents

1.1 Learning to generalize 1
1.2 Learning to act 5
1.3 Understanding the learning process 7
1.4 Thesis contributions and structure 9

1.1 Learning to generalize

The ability to generalize a lesson from the classroom to the real world is what separates *learning* from *memorization*. In a range of tasks ranging from mathematics to language, humans are remarkably skilled at identifying abstract patterns, and applying these patterns to novel contexts. The reader is unlikely to have previously encountered the sentence 'the armadillo tipped its blue hat and returned to the game of marbles', and yet would likely have no trouble interpreting it, or answering questions concerning the colour of the armadillo's hat. However, the range of domains in which humans exhibit this ability to learn and generalize is limited. A human can easily parse natural language, but will struggle to identify structure in strings of base pairs arising from a genome. In these settings, we can benefit from computational tools. Machine learning approaches, in particular the training of deep neural

networks on large datasets, present a promising direction towards the development of general algorithms which can identify patterns in data and solve a range of problems.

In a typical machine learning pipeline, the practitioner collects data (a *training set*) which is then fed into a machine learning algorithm, with the hope that a system which can accurately model this data will have captured the underlying structure of the task. The training set will not encompass the set of all possible inputs a model may receive at deployment; a learned predictor must *extrapolate* from the data it was trained on in order to make useful predictions. Generalization is crucial both to obtain good performance and to ensure the safety and reliability of these systems when they encounter data that was not seen during training. Yet how do we ensure that these highly expressive systems are learning and not memorizing? This question is a central concern of a long line of literature, and of this thesis.

1.1.1 Defining generalization

The machine learning community broadly distinguishes between two classes of generalization: within-distribution generalization, and out of distribution (OOD) generalization. While both types of generalization concern the performance of a predictor on data not seen during training, they differ in their structural assumptions on how this data is generated. Within-distribution generalization assumes that the process by which the training data was collected will also generate the data on which we will evaluate the trained model. This assumption is used in a rich theoretical literature which provides provable guarantees on the generalization performance of certain classes of learning algorithms. However, it is not reflected in many of the settings found in practice, where the procedure by which the training data is collected differs from how data is generated at evaluation; this difference is widely referred to as *distribution shift*.

The real world is full of nonstationarities which can induce distribution shift: a user's taste in films will evolve as they age, slang terms enter and leave common usage, and interest in advertisements for winter boots may vary with the seasons. Data collected one month may quickly fall out of date and cease to be reflective

of the phenomenon being modelled within a matter of weeks. Even the ways in which data is collected may introduce biases into the training set. Many image datasets, for example, include the main subject centred nicely in the middle of the frame, signalling to a learning algorithm that the corners of an image do not contain relevant information. These nonstationarities and biases can result in large distribution shifts when the model is deployed. The technical difficulty of overcoming such distribution shifts will depend on their structure and magnitude. Work on developing models which are robust to distribution shifts typically must make explicit assumptions on the type of shift being considered.

1.1.2 The importance of generalization

While we often marvel at great feats of memorization, such as memory champions who can recite the order of a shuffled deck of cards after a minute's concentration or taxi drivers who can recall the street maps of large cities by heart, memorization is discouraged when it occurs in place of understanding. The difference between the two is one of generalization: memorizing multiplication tables enables quick recall, but learning how to multiply numbers together algorithmically enables generalization to previously unseen number pairs. While we hesitate to anthropomorphize a linear regression model in saying that it 'understands' the relationship between inputs and outputs, we nonetheless seek out a similar phenomenon in our machine learning systems. It is not sufficient to make perfect predictions on the training data: the model must be able to apply the relationship between input and output to new contexts, and make accurate predictions in these contexts.

This generalization is crucial if we hope to see the power of machine learning deployed in real-world settings, where mistakes on novel inputs can have catastrophic consequences. Unforeseen distribution shifts in medical data, such as replacing an imaging device in a hospital, can result in a decline in performance that has serious ramifications for patients' health. Cross-validation, the canonical approach to evaluating generalization performance, will not identify the model's robustness to distribution shifts that the developer did not already foresee and include in

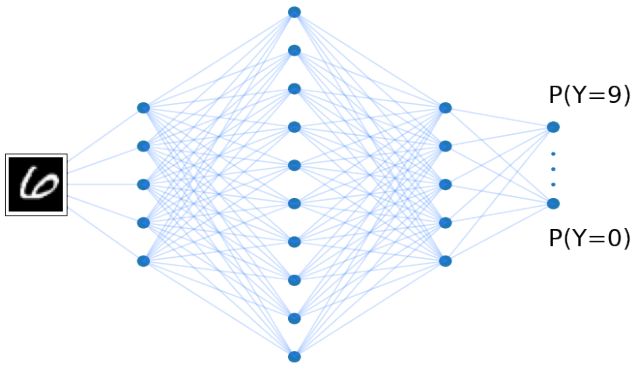


Figure 1.1: A visualization of a fully connected multi-layer perceptron. The neural network consists of a series of layers, where each layer applies a non-linear activation to a linear combination of activations from the previous layer. Each circle in this figure corresponds to an activation unit, while the lines correspond to weights connecting the left hand side layer to the right. The output of the network in this case corresponds to a probability distribution indicating the likelihood that the input image corresponds to a given digit.

the evaluation set. This presents a particular challenge for practitioners; even if a decline in performance is detected, the best we can hope to do is re-train or fine-tune the model to improve its performance on the new data. This can become much more expensive than if the model had learned relationships that generalized well off the bat.

1.1.3 Deep learning

The deep learning revolution has spurred the growth of several international conferences, the creation of multiple industrial AI labs, and the allocation of three Turing awards. This is due to the success of deep neural networks (DNNs, see Figure 1.1) at modelling a wide range of data modalities. The applications of DNNs range from the benign, such as helping people with visual impairments navigate a street and translating text from one language to another, to those with the potential for malicious use, such as identifying human faces in surveillance footage. Strikingly, DNNs often obtain impressive generalization performance and are considered robust enough to be used in many commercial applications.

The success of these models has brought attention to the discipline, but also stymied theoreticians. Compared to the expressivity of deep neural networks, traditional learning algorithms seek to model data by searching over a relatively small class of functions. Theoretical analysis of these algorithms crucially depends on the size of the function class in order to provide guarantees on the expected error of the function found by the algorithm on new data. In contrast, the set of functions expressible by a given neural network architecture is so large as to result in vacuous results when traditional analysis is applied to deep learning. This has opened up a number of exciting approaches to study the generalization of DNNs which often have a more experimental flavour than prior work on learning theory.

1.2 Learning to act

We will be particularly interested in studying machine learning systems which are capable of *doing things* in the world, rather than passively outputting predictions. This is captured by the reinforcement learning (RL) framework (see Figure 1.2). At its core, a reinforcement learning problem consists of an *agent* which can interact with an *environment* with the goal of maximizing the cumulative *reward* signal that it receives. Just as a trainer can teach a dog to sit by providing suitable rewards to reinforce the desired behaviour, we can apply the power of machine learning algorithms to maximize a prespecified reward function in the RL framework. We use the terminology *behaviour policy* to refer to the distribution over actions that the agent takes in each state of the environment. The *optimal policy* is the action-selection rule which maximizes the expected cumulative reward from each state.

1.2.1 The reinforcement learning problem

Learning how to behave optimally is often aided by learning to predict the expected cumulative reward the agent will receive after it visits a state and then follows some behaviour policy. Our usage of the word 'learn' differs from that used in other areas of machine learning, where it refers to the identification of a relationship between inputs and outputs from a data set. An RL agent does not receive explicit



Figure 1.2: A figure visualizing the RL problem; featuring the environment (pictured on the left) and agent (right) dichotomy.

information about the optimal policy from the environment; this policy must be deduced from the reward and transition structure via planning. Reinforcement learning is thus closely related the problem of *optimal control*. Control problems assume an environment, modelled as a Markov Decision Process (MDP), with known transition and reward structure and seek to identify an optimal behaviour policy. Reinforcement learning also seeks to obtain an optimal policy, but does not assume that the structure of the MDP is known in advance. Instead, the agent must interact with the MDP in order to obtain information about the reward and transition structure, and use this information to identify an approximately optimal policy. The number of interactions with the environment needed for an agent to identify an approximately optimal policy, its *sample complexity*, is a key criterion by which RL algorithms are evaluated, and which distinguishes reinforcement learning from optimal control, where the dynamics of the world are known a priori.

1.2.2 Generalization in reinforcement learning

Many problems of interest in reinforcement learning do not require generalization, i.e. it is assumed that the agent will only encounter states at deployment that it encountered during training. Indeed, the field contains a rich literature on the analysis of *tabular* problems, whereby the states of the MDP are simply an enumeration of integers and learning consists of updating a lookup table. In many *rich observation* settings, however, there may be an interesting functional

relationship between the state observations emitted by the environment and the reward and transition structure at that state. While lookup tables are sufficient for small state spaces, most applications of RL to real-world data necessitate generalization either due to the magnitude of the state space or in order to be robust to distribution shifts. For example, the angles and torques of a robot's actuators can take on a continuum of values, and the set of possible image inputs far exceeds the size that can be fit into computer memory as a lookup table. Further, even if such a representation of the value function were possible it is not clear whether that would be desirable. Visual similarity between states can provide information about the optimal policy and value function which may be useful to the agent. A function approximator with an appropriate inductive bias will be able to leverage this similarity to accelerate the learning process.

However, a dark side of generalization arises in reinforcement learning problems: instability. Instability is particularly problematic in some of the most popular algorithms in the deep RL literature, where careful hyper-parameter tuning and engineering tricks are needed to prevent the network parameters from diverging to infinite values. Excessive generalization can also slow down learning if the inductive bias encoded by the function approximator is not aligned with the structure of the environment. A bias towards smooth functions might thus result in a network that fails to accurately distinguish between states with large differences in value. The tension between generalization and stability in deep reinforcement learning presents an additional layer of difficulty to deep RL, as compared to supervised deep learning, and is a problem we will explore in later chapters.

1.3 Understanding the learning process

Throughout this thesis, we will seek to understand how a model will generalize by studying the optimization trajectory it took during training. In contrast, most theoretical results in the literature characterize generalization using only properties of the final outputs of a learning algorithm, i.e. the neural network's final trained parameters. Studying the trajectory of a learning algorithm (which we will refer

to as its *learning dynamics*) gives us the opportunity to gain insights into a model that cannot be obtained by only considering the final trained parameters. We will leverage these insights in later chapters to obtain novel estimators with significant predictive power over the ranking of a model's final generalization performance.

1.3.1 Learning to generalize between data points

Many neural network training procedures leverage large datasets which cannot fit onto a single GPU. To accelerate training, learning algorithms often partition the data into subsets called *minibatches*. The learner then updates its predictions for each minibatch iteratively. This procedure provides extremely useful information about whether the agent is learning to generalize or to memorize, by revealing whether the learner's update based on one minibatch has *generalized* to improve its predictions on the other minibatches.

A key intuition throughout this thesis is that generalization between disjoint subsets of the training set can be indicative of generalization to the test set. If an update to the network intended to improve its predictions for one minibatch also improves the network's predictions on many other data points, this is likely to result in an improvement to the learner's predictions on novel inputs drawn from the same process that generated the training data. In contrast, if an update does not improve the learner's predictions on the other data points in the training set, it is unlikely to improve the agent's predictions on the data it will see at deployment. This relationship will be explored in greater depth in Chapter 4.

1.3.2 Stability vs extrapolation

Our study of learning dynamics in RL agents will be particularly enlightening, as these dynamics are much more complex than their supervised counterparts. Supervised learning algorithms tend to induce well-behaved dynamics that, even if they correspond to a non-convex loss surface, nonetheless at least come with reasonable guarantees on the convergence of learning algorithms to local minima. Reinforcement learning, in contrast, incorporates nonstationarity as part of the

learning process. This nonstationarity has many forms: the distribution of states that the agent visits will change as its policy improves, but so will the target values that the agent is trying to predict. This results in a dynamical system that superficially resembles the stable gradient descent regime of supervised learning algorithms, but lacks convergence guarantees many problem settings of interest.

Even more pernicious, reinforcement learning agents must also face the challenge that the very properties of a function approximator which are associated with better generalization in the supervised learning setting are precisely those which can cause divergence in RL algorithms, as we will see in later sections. This means that deep RL agents must overcome two distinct hurdles in order to learn an optimal behaviour policy which also generalizes to new settings. First, they must learn to behave optimally in the training environment, while avoiding issues of divergence and other pathologies of function approximation in RL. Second, having achieved a high-performing policy, they must then ensure that the policy is robust to superficial changes to the observations they receive from the environment.

1.4 Thesis contributions and structure

1.4.1 Contributions

Broadly speaking, this thesis presents a set of novel empirical and theoretical tools to predict, understand, and improve generalization in deep neural networks in a range of problem settings. Crucial to these results will be an analysis of the *dynamics* of learning algorithms in various settings. The primary contributions of this thesis are enumerated as follows:

1. A characterization of the relationship between invariance, training speed, and generalization, with theoretical results complemented by empirical validation of our main findings in practically relevant settings.

    (a) A theoretical analysis of the effect of invariance on generalization via PAC-Bayes bounds, allowing an explicit characterization of the role of symmetries in generalization bounds via a quantity we term the

symmetrization gap. This theoretical analysis is complemented by an empirical study which highlights the limitations of the types of approximate invariance promoted by data augmentation to generalize to out of distribution inputs.

(b) A new estimator of the marginal likelihood which complements the analysis of upper bounds on generalization error to provide useful rankings of models for architecture search and hyperparameter selection. The analysis of this estimator reveals a deep connection between training speed and Bayesian model selection, and yields a novel performance estimator for architecture search in deep neural networks.

2. A theoretical framework for the study of representation dynamics in deep reinforcement learning along with practical insights derived thereof. These insights concern both the ability of a learned feature representation to linearly approximate a prediction target, and its ability to generalize to novel prediction objectives over the course of training.

   (a) A theoretical model and analysis of the representation learning dynamics of value-based reinforcement learning algorithms, yielding an analytic characterization of the effect of auxiliary tasks on agents' learned representations.

   (b) The identification of the 'capacity loss' phenomenon in deep RL which characterizes the tendency of neural networks to catastrophically overfit to early prediction targets in sparse-reward environments.

   (c) A new regularization method based on the insights from this previous analysis that improves generalization to new prediction objectives even after long training periods.

3. Theoretical analysis leveraging the above framework to provide insight into and algorithmic improvements to generalization to novel observations and environments.

(a) A theoretically grounded explanation for prior empirical observations of overfitting in the broader deep RL literature, including dense-reward problems, and a set of recommendations for principled approaches to reduce memorization and improve generalization between observations.

(b) A representation-learning objective which goes beyond the single-environment setting to enable zero-shot generalization to novel environments sharing underlying structure with the training environments.

1.4.2 Warmup: supervised learning

The first two content chapters of this thesis will lay the groundwork for our understanding of generalization in deep neural networks. Chapter 3 presents a novel PAC-Bayes generalization bound for invariant models, characterizing the effect of invariance on generalization through a quantity which we term the *symmetrization gap*. We empirically verify that the symmetrization gap appears in computations of PAC-Bayes bounds for invariant deep neural networks, and further show a strong correlation between the rankings of these upper bounds and the rankings given by final generalization performance. This chapter further contributes new empirical analysis of the optimization trajectories of DNNs trained to exhibit approximate invariances via data augmentation, illustrating the limitations of these trained invariances to extrapolate beyond their training distribution. The importance of invariance to generalization in reinforcement learning will be revisited in Chapter 8.

A failing of PAC-Bayesian generalization bounds is that they do not offer predictions about which of a set of models will generalize best. In order to achieve correctness, these bounds pay the price of predictive power. Chapter 4 shifts focus to consider performance estimators which can *predict* the relative performance of different neural network architectures. It presents a novel marginal likelihood estimator which can be used to enable Bayesian model selection for a broader class of models from which we only require accurate posterior samples, overcoming the normative limitations of generalization bounds and computational challenges of the exact marginal likelihood. This estimator has a number of appealing properties, chief

among which is that it can be applied to a subset of training losses from a gradient descent trajectory. This illustrates a deep relationship between generalization and learning dynamics, as a model's generalization performance can thus be said to depend on its *training speed*. These chapters will principally be based on the following papers:

- Clare Lyle, Lisa Schut, Robin Ru, Yarin Gal, and Mark van der Wilk. A bayesian perspective on training speed and model selection. *Advances in Neural Information Processing Systems*, 33, 2020
- Clare Lyle, Mark van der Wilk, Marta Kwiatkowska, Yarin Gal, and Benjamin Bloem-Reddy. On the benefits of invariance in neural networks. *NeurIPS Workshop on Machine Learning with Guarantees*, 2019b

with supporting evidence drawn from

- Binxin Ru, Clare Lyle, Lisa Schut, Mark van der Wilk, and Yarin Gal. Revisiting the train loss: an efficient performance estimator for neural architecture search. *Advances in Neural Information Processing Systems (Spotlight Presentation)*, 34, 2021

1.4.3 Generalization in reinforcement learning

The remaining chapters present an analysis of the learning dynamics of RL algorithms and explore a number of applications of this analysis to representation learning and generalization in deep RL. Chapter 5 lays down a novel *learning dynamics* framework which will be leveraged throughout the remainder of the thesis, and reveals that the dynamics followed by temporal difference methods can cause the learned representation to reflect aspects of the transition structure of the environment [Lyle et al., 2021a]. In Chapter 6, we explore one application of this analysis which yields a novel regularization approach, InFeR [Lyle et al., 2022a]. This method enables deep RL agents to attain nontrivial return in the notoriously difficult Montezuma's Revenge game using only a naive ϵ-greedy exploration algorithm. This section is based on the following papers:

- Clare Lyle, Mark Rowland, Georg Ostrovski, and Will Dabney. On the effect of auxiliary tasks on representation dynamics. In *Artificial Intelligence and Statistics (AISTATS)*, 2021a

- Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. In *International Conference on Learning Representations (Spotlight)*, 2022a

Chapter 7 explores implications of the previous chapters on generalization, providing novel analysis and insight into why value-based deep reinforcement learning often produces highly brittle agents [Lyle et al., 2022b], as well as identifying principled approaches to remedy the tendency of deep RL methods to overfit. Chapter 8 revisits the discussion of invariance from Chapter 3 with a new perspective grounded in causal inference, presenting a novel representation learning method to encourage generalization in some classes of MDPs by promoting invariance to spurious factors of variation in the environment [Zhang et al., 2020a, Lyle et al., 2021b]. These chapters are based on the following papers.

- Clare Lyle, Mark Rowland, Will Dabney, Marta Kwiatkowska, and Yarin Gal. Learning dynamics and generalization in reinforcement learning. *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022b

- Amy Zhang, Clare Lyle, Shagun Sodhani, Angelos Filos, Marta Kwiatkowska, Joelle Pineau, Yarin Gal, and Doina Precup. Invariant causal prediction for block MDPs. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020a

A number of papers that I worked on during my PhD did not fit into this thesis, including the following.

- B Wang, C Lyle, and M Kwiatkowska. Provable guarantees on the robustness of decision rules to causal interventions. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2021

- Angelos Filos, Clare Lyle, Yarin Gal, Sergey Levine, Natasha Jaques, and Greg Farquhar. Psiphi-learning: Reinforcement learning with demonstrations using successor features and inverse temporal difference learning. *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021

- Jannik Kossen, Neil Band, Clare Lyle, Aidan N Gomez, Tom Rainforth, and Yarin Gal. Self-attention between datapoints: Going beyond individual input-output pairs in deep learning. *Advances in Neural Information Processing Systems*, 2021

- Marc G Bellemare, Will Dabney, Robert Dadashi, Adrien Ali Taiga, Pablo Samuel Castro, Nicolas Le Roux, Dale Schuurmans, Tor Lattimore, and Clare Lyle. A geometric perspective on optimal representations for reinforcement learning. *Advances in neural information processing systems*, 2019

- Clare Lyle, Marc G Bellemare, and Pablo Samuel Castro. A comparative analysis of expected and distributional reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2019a

A discussion of contributions to joint work can be found at the end of the thesis. Appendices contain proofs, experiment details, and supplemental experimental results to those presented in each chapter.

The key idea driving this thesis is that properties of a network's training trajectory can tell us a great deal about how it will generalize to novel inputs. Chapters 3 and 4 ground this idea in the supervised learning regime by studying generalization between minibatches and validate its utility by developing both novel generalization bounds and practical model selection tools. Chapters 5 and 6 identify key properties of the training dynamics of reinforcement learning agents that differ from the supervised setting, and show how these properties can be both beneficial and detrimental to representation learning. Finally, Chapters 7 and 8 apply the notions of invariance and within-training-set generalization from Chapters 3 and 4 to the RL problem, leveraging the theoretical and empirical tools

presented in Chapters 5 and 6 to quantify memorization and improve generalization in reinforcement learning. We will conclude with a discussion of how these ideas can be (and in some cases have already been) further leveraged in the pursuit of learning systems which can effectively learn and generalize across a range of tasks.

2

Background & literature review

Contents

2.1 Learning frameworks 16
2.2 Generalization in supervised learning 24
2.3 Generalization in reinforcement learning 33

This chapter will provide the high-level background and literature necessary to contextualize the contributions of this thesis, and set out a standard set of notation that will be used in the following chapters. Where necessary, individual chapters may also contain a background section; these sections will relay information that pertains only to the contents of the chapter that contains them.

2.1 Learning frameworks

A learning algorithm can be applied to a wide variety of problems, and it is often useful to categorize learning problems based on the information available to the algorithm. We might task a learning system with identifying a mapping between input-label pairs (supervised learning), with constructing an embedding of inputs that captures relevant structure (unsupervised learning), with generating samples from some distribution (generative modelling), or with maximizing a reward

signal via interaction with an environment (reinforcement learning). The types of algorithms we can deploy in each of these situations differ not only in terms of the types of outputs they produce, but also in the stability of their learning dynamics. This thesis will focus primarily on the distinction between supervised learning, where learning dynamics of gradient descent algorithms are relatively straightforward to analyze, and reinforcement learning, wherein analogues of even simple methods such as linear regression can suffer from instability and divergence.

2.1.1 Supervised learning

Supervised learning is concerned with characterizing a functional relationship between inputs and labels. This framework assumes that the data takes the form of input-label pairs $(\mathbf{x}, y)$ generated by sampling from some distribution $P_{\mathcal{D}}$. The objective of the learning algorithm is to find a function f such that $f(\mathbf{x}) = y$. The task of finding such a function is nontrivial: one must both propose a suitable class of functions over which to search, and an effective means of identifying functions from this class which are likely to capture the target relationship.

Empirical risk minimization

We first consider the task of identifying a candidate function f with the property that $f(\mathbf{x}) = y$ on $(\mathbf{x}, y)$ pairs sampled from $P_{\mathcal{D}}$, including those not seen during training. We use a loss function ℓ to quantify the quality of $f(\mathbf{x})$ as an approximator to y; we will also refer to the expectation of this quantity as the *risk*. When y belongs to a continuous space, we call this a *regression* problem. When y belongs to a finite set, we have a *classification* problem.

For a class of functions $F = \{f : \mathcal{X} \to \mathcal{Y}\}$, a set $\mathcal{D}^n = (\mathbf{x}_i, y_i)_{i=1}^n \sim P_{\mathcal{D}}$, and a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, we define the expected risk R_ℓ as

$$R_\ell(f) = \mathbb{E}_{(\mathbf{X},Y)\sim P_{\mathcal{D}}}[\ell(f(\mathbf{X}), Y)] \, . \tag{2.1}$$

Similarly, we define the empirical risk $\widehat{R}_\ell$ as

$$\widehat{R}_\ell(f, \mathcal{D}^n) = \tfrac{1}{n} \textstyle\sum_{i=1}^n \ell(f(\mathbf{x}_i), y_i) \, . \tag{2.2}$$

In regression problems, ℓ is typically set to be the squared error $(f(\mathbf{x}) - y)^2$. In classification, it is usually the cross-entropy loss between a categorical distribution $p(\cdot|\mathbf{x})$ and the Dirac delta distribution at the label y. In this case, the hypothesis class F will consist of mappings from inputs to *distributions* over labels in $\mathcal{Y}$.

When the hypothesis class F contains the true functional relationship f, the learning problem is realizable. However, most settings of interest are not realizable, and so we seek instead a function f^* which minimizes the expected loss over the data-generating distribution, which can be expressed formally as follows.

$$f^* = \text{argmin}_{f \in F} R_\ell(f) = \text{argmin}_{f \in F} \mathbb{E}_{(\mathbf{X},Y)\sim P_\mathcal{D}}[\ell(f(\mathbf{X}), Y)] \tag{2.3}$$

In practice, computing this expectation is impossible and we instead use samples to estimate its true value. The empirical risk minimization framework assumes a finite sample $(\mathbf{x}_i, y_i)_{i=1}^n = \mathcal{D}^n \sim P_\mathcal{D}$, and proposes to find a function $f \in F$ that minimizes the empirical expectation of the loss over this sample, i.e. the empirical risk. Concretely, the function $\hat{f}$ is called the empirical risk minimizer if the following holds:

$$\hat{f} = \text{argmin}_{f \in F} \widehat{R}_\ell(f) = \text{argmin}_{f \in F} \frac{1}{n} \sum_{i=1}^{n} \ell(f(\mathbf{x}_i), y_i) \ . \tag{2.4}$$

The empirical risk minimization principle has seen widespread application in the machine learning literature [Vapnik, 1991, Donini et al., 2018]. Vapnik [1968] characterizes a number of appealing asymptotic properties of the empirical risk minimizer under certain conditions on the function class F; this analysis has formed the basis for the work on generalization bounds which will be discussed in Section 2.2.1. This principle is agnostic to the choice of function class, and does not give guidance on how to obtain a minimizer f^* when such classes are too large for exhaustive search. The following discussion will focus on one such class and search procedure: neural networks trained with gradient-based optimization.

Deep learning

Deep neural networks (DNNs) form a powerful and expressive class of function approximators [Raghu et al., 2017]. A DNN is a parameterized function f_θ which consists of layer-wise computations going from input to output. Some neural architectures include recurrent connections, where the output of the network is fed back into itself as input [Hochreiter and Schmidhuber, 1997b]; this thesis will focus exclusively on feedforward architectures, where only a single pass through the network is executed in order to obtain the function outputs. Feedforward neural networks constitute a rich and widely-used class of models whose dynamics are more amenable to analysis, including fully-connected multi-layer perceptrons (MLPs), convolutional neural networks, transformers [Vaswani et al., 2017], and ResNets [He et al., 2016]. The function f^k computed by each layer k of a feedforward neural network typically consists of a linear transformation of the output of the previous layer, followed by a non-linear activation function. A variety of activations have been used in DNNs; one popular example is the Rectified Linear Unit (ReLU), of the form $\sigma(x) = \max(0, x)$. The output of a neural network can thus be expressed as a composition of layer-wise operations,

$$f_\theta(\mathbf{x}) = f_\theta^L \circ \cdots \circ f_\theta^1(\mathbf{x}) \ . \tag{2.5}$$

Deep neural networks are trained using gradient-based optimization algorithms. The most fundamental of these is *stochastic gradient descent*. In this setting, the data $\mathcal{D}^n$ is uniformly at random divided into minibatches of size k, $((\mathbf{x}_{b_1}, y_{b_1}), \ldots, (\mathbf{x}_{b_k}, y_{b_k}))_{b=1}^{\lfloor n/k \rfloor}$. At each minibatch, we compute the gradient of the loss to obtain an update direction g as follows,

$$g(\theta, \mathcal{D}_b) = -\nabla_\theta \frac{1}{k} \sum_{i=1}^{k} \ell(f_\theta(\mathbf{x}_{b_i}), y_{b_i}) \ .$$

This yields an iterative algorithm where the parameters θ are updated according to the gradient for each successive minibatch. In most cases, we use a step-size

parameter $\alpha \in (0, 1]$ to improve the stability and convergence properties of the algorithm. The iteration step typically takes the following general form:

$$\theta_{t+1} \leftarrow \theta_t + \alpha_t g(\theta_t, \mathcal{D}_t) \ . \tag{2.6}$$

Many formulations of gradient descent allow the step size to depend on the iteration t; such dependence on the iteration is crucial to obtain convergence guarantees [Robbins and Monro, 1951]. Many adaptive optimization schemes [Duchi et al., 2011, Kingma and Ba, 2015] further accumulate parameter-dependent learning rates, and allow the update direction $g(\theta_t, \mathcal{D}_t)$ to also depend on prior gradients. While these optimizers will feature in the empirical analysis that appears later, their precise form is not important to our discussion.

2.1.2 Reinforcement learning

Whereas the supervised learning framework seeks to model a functional relationship between two variables, the reinforcement learning framework models an agent's interaction with an environment with the goal of identifying a behaviour policy which maximizes some reward signal. We model the environment as a Markov Decision Process (MDP) $\mathcal{M} = (\mathcal{X}, \mathcal{A}, R, P, \gamma)$, where $\mathcal{X}$ denotes the state space, $\mathcal{A}$ the action space, $R : \mathcal{X} \to \mathbb{R}$ the reward function, $P : \mathcal{X} \times \mathcal{A} \to \mathscr{P}(\mathcal{X})$ the transition probability function, and γ the discount factor. The agent obtains observation x which indicates the environment's state. It may then take an action $a \in \mathcal{A}$, after which the environment outputs a new observation x' and a reward r. The agent's objective is to maximize the cumulative discounted reward it receives over time. In *finite-horizon* environments, the agent may only take a finite number of steps; in *continuing* or *infinite-horizon* environments, on which this thesis will predominantly focus, the agent may take an unlimited number of steps in the environment. The discount factor γ determines the degree to which near-term rewards are preferred, resulting in the maximization target $\sum_{k=0}^{\infty} \gamma^k R_k$, called the *return*. The return is a random variable that depends on the sequence of states and actions taken by the agent. The value of a state-action pair under some action-selection policy

$\pi : \mathcal{X} \to \mathscr{P}(\mathcal{A})$ is equal to the expected value of the return starting at some state-action pair (x, a) and following the policy π. This can be expressed by the action-value function $Q^\pi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, defined as

$$Q^\pi(x, a) = \mathbb{E}_{\pi,\mathcal{P}}[\sum_{k=0}^{\infty} \gamma^k R(x_k, a_k) | x_0 = x, a_0 = a] \,. \tag{2.7}$$

Value-based methods seek to learn the (resp. action-) value function $V^\pi : \mathcal{X} \to \mathbb{R}$ (resp. $Q^\pi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$) associated with some policy π [Sutton and Barto, 2018]. In particular, we are interested in learning the value function associated with the optimal policy π^* which maximizes the expected discounted sum of rewards from any state. Such a value function is then straightforward to translate into an optimal behaviour policy: the agent need only take the action with the highest predicted value at each state.

At the core of value-based RL are the *Bellman operators* [Puterman, 1994]. The Bellman policy evaluation operator $T^\pi : \mathbb{R}^\mathcal{X} \to \mathbb{R}^\mathcal{X}$ is defined with respect to a policy π and provides a method to *update* a predicted value function V to more closely resemble the value V^π of the policy π. It is defined as

$$(T^\pi V)(x) = \mathbb{E}_{X_1 \sim P(\cdot|x,\pi(x))}[R(x) + \gamma V(X_1)] \,.$$

Introducing a matrix notation of the transition operator $P^\pi \in \mathbb{R}^{\mathcal{X}\times\mathcal{X}}$ defined by $P^\pi[x, x'] = \sum_{a\in\mathcal{A}} \pi(a|x) P(x'|x, a)$, and the expected reward vector $R^\pi \in \mathbb{R}^\mathcal{X}$ defined by $R^\pi(x) = \mathbb{E}_\pi[R_0|X_0 = x]$, this can be expressed even more succinctly in matrix notation as

$$T^\pi V = R^\pi + \gamma P^\pi V \,.$$

T^π is a contraction on the space of value functions [Puterman, 1994], and so repeated application of T^π to any initial value function converges to V^π [Bertsekas and Tsitsiklis, 1996]. For control problems, where we seek to obtain the value of an unknown optimal policy, learning requires not only estimating the value of a policy but also improving that policy to increase its expected return. In this

setting, we leverage an analogous operator termed the Bellman optimality operator $T^* : \mathbb{R}^{\mathcal{X}\times\mathcal{A}} \to \mathbb{R}^{\mathcal{X}\times\mathcal{A}}$. The action of T^* on value functions is defined by

$$(T^*Q)(x,a) = \mathbb{E}_{x'\sim P(x,a)}[R(x) + \gamma \max_{a'\in\mathcal{A}} Q(x',a')] .$$

The Bellman optimality operator attains similar convergence guarantees as the policy evaluation operator, and can be shown in tabular state spaces to converge to the value of the optimal policy. In principle both T^* and T^π can be defined over action-value or state-value functions, but this thesis will predominantly consider the application of T^π to value functions and T^* to action-value functions, a choice reflected in the notation of this chapter. We will refer to the value $T^\pi V$ or T^*Q as the *Bellman target* associated with the (resp. action-) value function V (resp. Q), where the operator in use will be clear from context.

In most settings of interest, we do not have access to the expected reward vector R^π or the environment transition matrix P. As a result, value-based RL agents must use sampled transitions of the form $(x_t, a_t, r_t, x_{t+1}, a_{t+1})$ to approximate these updates. In the case of the policy evaluation operator, this sample-based approximation takes the form of the SARSA update, so called due to its use of (**S**tate-**A**ction-**R**eward-**S**tate-**A**ction) transitions. The SARSA algorithm assumes that the transition has been sampled from some fixed behaviour policy π, and estimates Q^π by iteratively applying the update rule

$$Q_{t+1}(x_t,a_t) = Q_t(x_t,a_t) + \alpha[r_t + \gamma Q_t(x_{t+1},a_{t+1}) - Q_t(x_t,a_t)] . \tag{2.8}$$

Sample-based methods will use some step size $0 < \alpha < 1$ in order to average out noise in the Bellman targets due to stochasticity in the environment. The seminal Q-learning algorithm [Watkins and Dayan, 1992], which forms the basis of many deep RL agents [Mnih et al., 2015], can similarly be viewed as approximating the iterative application of T^* and related operators [Tsitsiklis, 1994, Jaakkola et al., 1994, Bertsekas and Tsitsiklis, 1996]. Q-learning is based on the following update

$$Q_{t+1}(x_t,a_t) = Q_t(x_t,a_t) + \alpha[r_t + \gamma \max_{a'\in\mathcal{A}} Q_t(x_{t+1},a') - Q_t(x_t,a_t)] . \tag{2.9}$$

Policy gradient methods [Sutton et al., 2000] operate directly on a parameterized policy π_θ. Rather than first going to the trouble of learning a value function, policy gradient methods directly optimize the parameters θ of the policy so as to maximize the expected return $J(\pi_\theta) = \mathbb{E}_{s_0 \sim P_{\mathcal{M}}(s_0)} V^{\pi_\theta}(s_0)$. The gradient of this loss can be estimated from sampled trajectories when expressed as follows (c.f. [Sutton and Barto, 2018] for a more detailed derivation and exposition):

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{x_t, a_t \sim P(\cdot|\pi_\theta)}[\nabla_\theta \log \pi_\theta(a_t|x_t) Q^{\pi_\theta}(x_t, a_t)] \ . \tag{2.10}$$

Variations on this learning rule include *actor-critic* methods [Konda and Tsitsiklis, 2000], which use a baseline given by a value-based learner to reduce update variance, and trust-region based methods, such as Trust Region Policy Optimization [Schulman et al., 2015] and Proximal Policy Optimization (PPO) [Schulman et al., 2017].

Function approximation schemes enable RL agents to generalize their knowledge about the value or policy at one state to other states in the environment. Linear function approximation assumes state-action pairs are embedded as features $\phi(x, a) \in \mathbb{R}^d$, and some linear map $\mathbf{w} \in \mathbb{R}^d$ is used to approximate the value function $Q^\pi(x, a) = \langle \phi(x, a), \mathbf{w} \rangle$. This regime has been the study of a rich literature exploring the stability and sample complexity of RL in the presence of function approximation [Jin et al., 2020, Wang et al., 2020, Precup et al., 2001, Tsitsiklis and Van Roy, 1996].

A second, more widely-used family of algorithms involve the use of neural networks as function approximators of the form $Q_\theta : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$. This is the *deep RL* regime. This approach is well-suited to an array of complex tasks, ranging from systems control problems [Degrave et al., 2022] to video games [Mnih et al., 2015]. At its core, value-based deep RL involves training a neural network to approximate the Bellman targets of a predicted value function using (semi-, see e.g. [Sutton and Barto, 2018]) gradient descent. This approach computes a semi-gradient update direction $f(\theta)$ of the following form, where $a^* = \max_a(Q_\theta(x_{t+1}, a))$.

$$f(\theta) = (\nabla_\theta Q_\theta)[r_t + \gamma Q_\theta(x_{t+1}, a^*) - Q_\theta(x_t, a_t)] \tag{2.11}$$

Because the final layer of a neural network is usually linear, it is possible to express the output $Q_\theta(x, a)$ in the form $Q_\theta(x, a) = \langle \phi_{\theta'}(x, a), \mathbf{w} \rangle$, and $\theta = \theta' \oplus \mathbf{w}$ where $\oplus$

denotes concatenation. Under this parameterization, the map $\phi_{\theta'}$ is referred to as the *feature map*. This framework has been used to study the learned representations of deep reinforcement learning agents in a number of recent works [Kumar et al., 2021, Lan et al., 2022, Lyle et al., 2019a].

2.2 Generalization in supervised learning

We now turn our attention to the principal object of interest in this thesis: generalization. The study of generalization in supervised learning problems spans decades, from the seminal work of Vapnik [1968] to recent exciting developments in the kernel analysis of deep networks [Jacot et al., 2018] and the interpolation regime [Bartlett et al., 2020]. We will begin by presenting classical bounds on the generalization error of a learning algorithm's output. However, these classical approaches, based on quantifying the complexity of an algorithm's hypothesis class, fail to account for the generalization performance of deep neural networks, motivating more recent empirical approaches. We will conclude with an overview of the current state of the art of our understanding of generalization in deep learning.

2.2.1 Generalization bounds

The empirical risk minimization (ERM) framework puts forward the maxim: 'always pick a hypothesis which minimizes the risk on the training set'. This hypothesis will not in general attain the lowest risk on the underlying data-generating distribution, however. A long line of work [Vapnik, 1968, 1999, Bousquet and Elisseeff, 2002] has characterized upper bounds on the gap between the empirical risk of a hypothesis and its true risk, yielding *generalization bounds* which provide high-probability guarantees on the expected risk of a hypothesis. Of particular interest to us will be the application of such bounds to hypothesis classes generated by neural networks [Baum and Haussler, 1989, Dziugaite and Roy, 2017, Bartlett et al., 2017]. While generalization bounds for any hypothesis class are almost always looser than the upper bound on the expected risk given by computing the model's validation loss on held-out data, the study of generalization bounds continues

to thrive as a means of developing theoretical insight into learning algorithms, motivating their inclusion in our discussion.

Formalism

Most generalization bounds in the literature share a similar structure, consisting of the sum of a hypothesis' empirical risk and a complexity measure scaled by a function (typically the inverse square root) of the number of samples. We use the notation of Section 2.1.1, and let f be a function output by some learning algorithm with access to data $\mathcal{D}^n$ of size n, drawn from hypothesis class F. We let R_ℓ and $\widehat{R}_\ell$ be defined as in Equations 2.1 and 2.2 respectively. Letting $C(F)$ denote some complexity measure (for example, the logarithm of the number of functions in the hypothesis class) and $g(\cdot)$ some non-negative function defined over $\mathbb{R}$, usually $g(x) = \log(\frac{1}{x})$ or something similar, we obtain the generic form

$$R_\ell(f) \leq \widehat{R}_\ell(f) + \sqrt{\frac{C(F) + g(\delta)}{n}} \text{ with probability } 1 - \delta. \tag{2.12}$$

The magnitude of the complexity measure $C(F)$ is crucial to the tightness of a bound. In some cases the complexity measure which bounds the expected risk may take a value so large that it dwarfs the maximal value the risk can obtain, resulting in bounds that are *vacuous*. For example, a vacuous bound would guarantee that a neural network's probability of making a classification error on new samples from the the MNIST dataset will be less than 500%. In neural networks, bounds based on the margin around the decision boundary [Wei and Ma, 2019], the VC dimension [Harvey et al., 2017], and the spectral norm of the network weights [Bartlett et al., 2017] all become vacuous for network architectures of the scale typically applied to popular benchmarks such as CIFAR-10 or ImageNet [Bartlett, 1997, Dziugaite and Roy, 2017]. However, recent work has found complexity measures which *can* capture a reasonable notion of simplicity on neural networks – at least to the point where bounds are non-vacuous [Dziugaite and Roy, 2017].

Complexity and Occam's razor

The term $C(F)$ can be interpreted as a form of Occam's razor applied to the hypothesis class: given hypotheses drawn from two classes which attain the same empirical risk, we should prefer the hypothesis drawn from the simpler class. This simple idea drives most results in model selection and generalization in machine learning [Rasmussen and Ghahramani, 2001]. However, while generalization bounds of the flavour shown above characterize the complexity of the entire class of hypotheses, one might also prefer to use Occam's razor as a tool to select hypotheses within a single class. This is the core idea behind PAC-Bayes generalization bounds [McAllester, 1999, Langford and Shawe-Taylor, 2003, Lever et al., 2013, Catoni, 2007], which allow us to assign a complexity penalty that distinguishes between different hypotheses within a function class for randomized predictors. Leveraging a notion of hypothesis-level complexity is a powerful tool [Bartlett, 1997], yielding some of the first non-vacuous generalization bounds for over-parameterized neural networks [Dziugaite and Roy, 2017].

Occam's razor does not give us an out-of-the-box definition of simplicity, however. Two such definitions are favoured by the machine learning community to explain generalization of deep networks. The first of these is flatness of the local minima to which gradient-based optimization tends to converge [Hochreiter and Schmidhuber, 1997a]. Much of the folk wisdom surrounding generalization in deep learning hypothesizes that the reason neural networks generalize is because stochastic gradient descent drives the learned weights towards flat regions of the loss landscape [Keskar et al., 2017]. This picture is slightly complicated by the observation of Dinh et al. [2017] showing that sharp minima can also generalize, but nonetheless demonstrates strong predictive power in empirical evaluations [Jiang et al., 2020]. Optimization algorithms which deliberately add noise to the training process, such as entropy-SGD and the direct optimization of a PAC-Bayes bound [Chaudhari et al., 2017, Dziugaite and Roy, 2018a,b], have been shown to increase flatness and

improve the tightness of some generalization bounds on neural networks. PAC-Bayes risk bounds are also deeply connected to the Bayesian notion of marginal likelihood [Germain et al., 2016a].

A second widely-used notion of simplicity is model compressibility, whereby models which can be compressed to a smaller length are said to be simpler. This is also referred to as the minimum description length (MDL) principle [Akaike, 1998, Hinton and Van Camp, 1993]. Compressibility and flatness overlap significantly: parameters in a flat region of the loss landscape can be perturbed, for example by a quantization step of a compression algorithm, without significantly affecting the loss. However, the techniques used to measure the two quantities differ, with compression approaches typically being more compute-intensive [Zhou et al., 2019, Ullrich et al., 2017]. PAC-Bayes bounds are appealing as either the flatness of minima [Neyshabur et al., 2019, 2017] or the compressibility [Zhou et al., 2019] notion of simplicity can be used to define the model complexity term in the bound.

The overparameterized regime

At the core of the challenge of applying results from statistical learning theory to neural networks is the expressiveness of neural network function classes; the overparameterized models of recent years are capable of memorizing even uniform random labels of the data [Zhang et al., 2017]. Uniform convergence bounds which depend on worst-case analysis are thus limited in the guarantees they can provide these networks. Indeed, Nagarajan and Kolter [2019] argue that certain forms of uniform convergence results may be fundamentally incapable of explaining generalization in deep learning, presenting a simple example of a learning problem where it is impossible to use uniform convergence guarantees to characterize the generalization performance of a neural network function class. While Negrea et al. [2020] and Bartlett et al. [2020] show that modified analysis can still yield uniform convergence results for overparameterized predictors in similar contexts, these results require studying either a surrogate predictor, or a restricted problem setting.

Even more intriguing has been the observation that, contrary to the received wisdom in learning theory that overparameterized models will overfit to their training data, increasing overparameterization can lead to *improved* generalization [Neyshabur et al., 2015a]. These empirical observations have prompted the study of interpolating predictors [Bartlett et al., 2020], which seeks to outline conditions by which a model which can attain zero empirical risk may still obtain near-optimal risk or robustness properties on the data-generating distribution [Bubeck and Sellke, 2021, Koehler et al., 2021]. This work is intricately connected to the double descent phenomenon, whereby the risk of a model, when plotted against the number of parameters, exhibits two descent regions: one in the underparameterized regime, and one in the overparameterized regime [Advani et al., 2020, Belkin et al., 2019, Nakkiran et al., 2021].

A gap remains, however, in leveraging these insights to attain tight bounds on the generalization error of modern neural network architectures trained on real-world datasets. Even worse, Jiang et al. [2020] show that many of the complexity measures that appear in generalization bounds are *negatively* correlated with generalization. This presents a double blow to the argument that generalization bounds might provide insight into what makes neural networks generalize well: not only are such bounds too loose to give practically relevant information, but they do not even offer an accurate *ranking* of models.

2.2.2 Explaining generalization without bounds

Why does a given neural network generalize well or poorly? This is a question that generalization bounds cannot (currently) answer, yet is crucial to the principled development and application of neural networks to real-world datasets. A large value of a complexity measure does not entail that a learning algorithm will not generalize; it simply states that there is not sufficient information to guarantee a small generalization gap. To address questions of *why* particular models generalize well, we must look to the particulars of the network architectures, training procedures, and datasets that arise from the natural world. The works discussed in this

section will present such an empirical study. While sometimes similar notions of complexity to those leveraged in generalization bounds may be used, the philosophical underpinnings of this research differ fundamentally from the formal guarantees of learning theory.

Scientific vs mathematical rigor

The scientific study of any phenomenon depends on a mixture of empirical experiments, during which data is gathered, and theory-building, where an explanation of the phenomenon is proposed [Popper, 1968]. A scientific theory is one that makes falsifiable predictions, which can then be tested by experiments. The generalization bounds of Section 2.2.1 can be viewed as theory-building in a loose sense; however, the goal of a generalization bound is to make a statement that will be *guaranteed* to hold with high probability, accepting that this may result in a large gap between the predicted upper bound and the empirical realization of the generalization error in some settings. This runs counter to the qualities of a good scientific theory, whose goal is to explain a phenomenon with as simple a model as possible, and to expose itself to the risk of falsification in doing so.

One notable step towards this framework of formulating and falsifying hypotheses arises from the work of Jiang et al. [2020] and Dziugaite et al. [2020], who conduct a large-scale empirical study of the correlation between *complexity measures* and generalization in deep neural networks. This line of work seeks to translate the formal guarantees of generalization bounds into testable scientific theories which make predictions about which models will generalize best. The key to doing so is to treat the ranking over models given by a complexity measure as a *prediction*, rather than an upper bound. If the generalization measure C is higher for network A than for network B, we interpret this as a prediction that network A should generalize worse than network B stemming from the theory that a model which generalizes well must do so *because* C is low.

Dziugaite et al. [2020] focus on identifying experimental settings where a measure *fails* to predict generalization, seeking to identify instances where the theory

can be falsified. Both works show that complexity measures based on flatness, such as PAC-Bayes generalization bounds [Dziugaite and Roy, 2017], are highly predictive of generalization across a range of experimental settings. However, none of the generalization measures studied by Dziugaite et al. [2020] robustly predicts generalization in neural networks under all experimental conditions, motivating further investigation of quantities which can predict – and ideally also explain – generalization in neural networks.

The loss landscape of neural networks

Much theoretical work studying neural networks has considered questions on the *existence* of parameters that allow the network to represent a function [Hornik et al., 1989, Raghu et al., 2017, Dong et al., 2020], but less attention has been paid to the search process by which such parameters might be found. To this end, recent work studying the loss landscape of neural network seeks to understand the properties of the optimization problem faced by neural networks in practice. These results have found that the initialization schemes used today [He et al., 2015, Glorot and Bengio, 2010] push the parameters towards a well-behaved region of the loss landscape that is relatively convex [Fort and Scherlis, 2019] and induces stable learning dynamics in suitable architectures [Yang and Schoenholz, 2017]. Indeed, many architecture design choices such as residual connections [He et al., 2016] and batch normalization [Ioffe and Szegedy, 2015] that improve performance can be shown to increase the smoothness of the loss landscape [Li et al., 2018, Santurkar et al., 2018], leading to minima which are flatter and should thus generalize better.

Much recent work has focused in particular on the geometry of the loss landscape around the minima found by gradient descent [Maddox et al., 2020]. One intriguing property identified recently is that of *linear mode connectivity*, whereby the convex combination of two locally optimal parameter vectors attains a similarly low loss as the two minima from which it was derived [Benton et al., 2021]. Frankle et al. [2020a] show that stochasticity in the optimization process *early* in training leads the trajectory towards disconnected regions of parameter space, while stochasticity

late in training results in a perturbation within a linearly connected loss basin. This geometric perspective is consistent with prior work studying the importance of the early learning period in determining the network's ability to represent certain functions [Achille et al., 2018]. Flatness properties of local minima have also been studied in the context of Bayesian deep learning [Izmailov et al., 2018], highlighting the connection between flat minima and approximations of the Bayesian marginal likelihood [Daxberger et al., 2021].

Correlates of generalization in deep learning

Much empirical work on generalization in deep learning seeks to identify measurable quantities which correlate with generalization in neural networks [Neyshabur et al., 2017]. This work has found such quantities in the robustness of minima to perturbations [Keskar et al., 2017] and to pruning [Bartoldson et al., 2020]. In a related work, Morcos et al. [2018] identify a relationship between the dependence of a network on single dimensions of activation space and generalization. Neyshabur et al. [2015b] propose that implicit norm-based regularization may contribute to the generalization performance of deep networks; some evidence suggests, however, that the mechanism by which overparameterization regularizes norm may not apply to fully general problem constructions [Hanin and Rolnick, 2019].

Further attention has been paid to regularization methods which improve generalization such as dropout [Srivastava et al., 2014], data augmentation [Wu et al., 2020], batch normalization [Ioffe and Szegedy, 2015], and weight decay [Arpit et al., 2017]. The study of early stopping [Duvenaud et al., 2016] has further revealed intriguing connections between variational inference and flatness of the loss landscape. However, the precise relationship between training speed, early stopping, and generalization has yet to be fully understood: Hardt et al. [2016] provide a theoretical and empirical analysis of the relationship between training time and generalization error; in contrast, Hoffer et al. [2017] observe that longer training can improve regularization provided a suitable optimization procedure is used. This is emblematic of a broader lack of granular understanding of generalization

in deep neural networks. While many correlates of generalization and techniques to improve generalization are widely used, the precise mechanisms by which they improve generalization have yet to be elucidated.

Studying the learning trajectory

The study of how networks evolve over the course of training has revealed a number of intriguing insights which promise to shed light on some of these mechanisms. A number of works have endeavoured to explain the efficacy of early stopping by arguing that neural networks learn functions of increasing complexity over the course of training [Rahaman et al., 2019, Arpit et al., 2017, Kalimeris et al., 2019], with similar biases towards other notions of simplicity also widely observed in the mapping between parameters and the resulting output functions of neural networks [Valle-Pérez et al., 2018, De Palma et al., 2019].

Approximating a discrete gradient descent trajectory as a continuous-time process in the limit of infinite layer width has facilitated closed-form analysis of the trajectory of deep neural networks [Jacot et al., 2018, Lee et al., 2019], which has sparked a number of intriguing insights into generalization [Arora et al., 2019], posterior sampling [He et al., 2020], and feature learning [Yang and Hu, 2021]. The analysis of Jacot et al. [2018] allows the optimization trajectory of a neural network to be modelled by a kernel gradient descent procedure with respect to a specific kernel called the neural tangent kernel (NTK). Smith et al. [2020] and Barrett and Dherin [2021] analyze a similar continuous-time approximation of gradient descent to identify an explicit regularization term that biases gradient descent algorithms towards flatter regions of the parameter space, providing insight into the relationship between the flatness of the region of the loss landscape traversed by gradient descent and sources of stochasticity such as minibatch sizes and finite learning rates.

2.2.3 Summary

As this section has shown, the question of why neural networks generalize has sparked a wide-ranging literature characterizing generalization and optimization

dynamics in DNNs. In spite of this rich literature, the answer to this question remains open. Generalization bounds present a domain of great theoretical interest, but fail to *explain* generalization in DNNs. Meanwhile, empirical work has identified many quantities corresponding to model complexity terms in generalization bounds that correlate with generalization [Jiang et al., 2020]. However, no single quantity studied thus far has been found to predict generalization performance under *all* possible experimental conditions [Dziugaite et al., 2020]. Complementary lines of work have shed significant insight into the optimization landscape and generalization properties of DNNs, but this picture is far from complete. A number of recent theoretical results, such as the analytic form of optimization dynamics in the infinite-width limit of DNNs [Jacot et al., 2018], the double descent phenomenon [Belkin et al., 2019], and benign overfitting [Bartlett et al., 2020], hint at a theoretical explanation of the benefits of overparameterization for generalization, but these findings do not currently apply to modern deep learning training regimes. Similarly, empirical investigation into the loss landscapes and gradient structure in DNNs has yielded principled insights into the optimization dynamics of deep networks, but still admits a gap between theory and practice. One contribution of this thesis will be to propose a family of performance estimators based on a network's training speed which are predictive of generalization performance rankings in a number of experimental settings. These estimators will be motivated by a theoretical and empirical analysis linking generalization bounds, invariance, and training speed, providing a potential mechanism to explain their success.

2.3 Generalization in reinforcement learning

The supervised learning regime has proven to be fertile ground for the study of generalization. However, reinforcement learning agents also benefit from accurate extrapolation to unseen inputs, and moreover existing approaches are particularly prone to overfitting [Zhang et al., 2018c]. Though many of the subproblems of generalization in reinforcement learning mirror those found in supervised learning

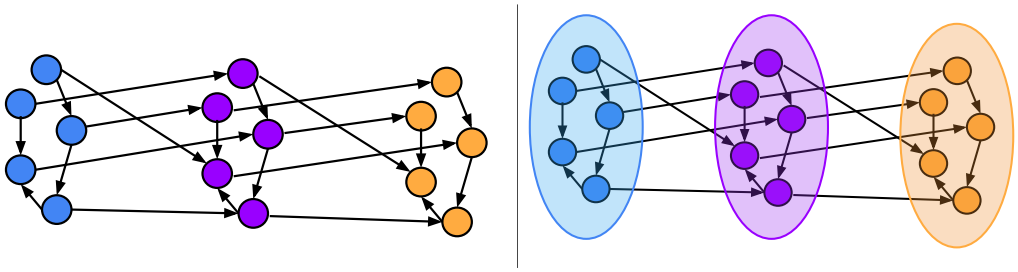


Figure 2.1: State abstractions allow an agent to treat similar states as though they were equivalent, potentially simplifying planning and generalization. Here, we see an example of grouping together states with similar transition dynamics, reducing the size of the state space from 15 states to 3.

[Cobbe et al., 2019], the nonstationarity of the training objectives used in deep RL adds additional challenges [Igl et al., 2021] which we will discuss in this section.

2.3.1 Single environment

Generalization is not always necessary for reinforcement learning: in small state spaces, for example, one may assume that all states the agent will encounter are known in advance and that the value function can be represented as a lookup table; the scope for generalization in this setting is limited to the context of learning an optimal policy under new reward functions [Dayan, 1993]. However, in large spaces generalization is crucial for sample efficiency, enabling faster convergence to an optimal policy and, in cases where the state space is so large that not all states will be visited during training, ensuring robust performance in previously-unseen states at evaluation time. This thesis will explore two different perspectives on generalization in a single environment: *state abstractions*, which can be leveraged to develop theoretical insights into the complexity of certain classes of environments, and *interference*, which studies how deep neural networks disentangle and distinguish between states.

State abstractions

State abstractions are used in reinforcement learning to simplify planning and to enable generalization in large or infinite state spaces; see Figure 2.1 for an illustration. Mathematically, a state abstraction is a mapping $\phi : \mathcal{X} \to \bar{\mathcal{X}}$, where

$\bar{\mathcal{X}}$ is some simplified space (for example, $\mathbb{R}^d$). A state abstraction is useful for reinforcement learning if it captures the similarity between states in the environment. That is, $\phi(s) = \phi(s')$ if and only if s and s' have similar values or optimal policies. Li et al. [2006] characterize a hierarchy of such abstractions, with the finest being a bisimulation of the original MDP and the coarsest being a partition of states according to the optimal action at each state. We will be particularly interested in the former, as our study of representation learning in Chapter 8 will focus on this class of state abstractions, known as *model-irrelevance* state abstractions. The following definition is for finite state spaces, but can be generalized to continuous state spaces.

Definition 2.3.1. *A* model-irrelevance state abstraction *on a Markov Decision Process* $\mathcal{M} = (\mathcal{X}, \mathcal{A}, R, P, \gamma)$ *is a mapping* $\phi : \mathcal{X} \to \bar{\mathcal{X}}$ *such that if* $\phi(x_1) = \phi(x_2) = \bar{x}$*, the following holds, where* $x' \in \mathcal{X}$*, and* $\bar{x}' \in \bar{\mathcal{X}}$*.*

1. $R(x_1, a) = R(x_2, a)\ \forall a \in \mathcal{A}$
2. $\sum_{x' \in \phi^{-1}(\bar{x}')} P(x'|x_1, a) = \sum_{x' \in \phi^{-1}(\bar{x}')} P(x'|x_2, a),\ \forall a \in \mathcal{A}$

There are a range of approaches to discovering and selecting state abstractions: Jong and Stone [2005] eliminate irrelevant variables from inputs where $\mathcal{X} \subset \mathbb{R}^n$; other approaches leverage statistical [Jiang et al., 2015] and information-theoretic [Abel et al., 2019] tools to aggregate states, while Taylor et al. [2008] construct state abstractions based on state similarity in the MDP. State aggregation is further leveraged in algorithms for RL in rich-observation settings [Misra et al., 2020].

In deep reinforcement learning, the state abstraction framework begins to merge with the feature-learning and representation-learning frameworks of deep learning. One can interpret the latent feature representation of the neural network as a state abstraction, and the remaining layers as a value function approximator. Gelada et al. [2019] and Zhang et al. [2020b] use ideas from the state abstraction literature to construct representation-learning objectives for DNNs which allow them to capture the structure of the environment. Such structure can be made explicit in the architecture of the neural network used as a function approximator to obtain

guarantees on the equivariance of the learned value function to symmetries that arise in the MDP [van der Pol et al., 2020b,a].

Representations in deep RL

A major challenge present in deep reinforcement learning is the sparsity of the reward signal relative to the rich structure of the environment. A broad range of auxiliary tasks have been developed over the years to overcome this sparsity in order to improve generalization and representation learning in deep RL [Jaderberg et al., 2017, Veeriah et al., 2019, Gelada et al., 2019, Machado et al., 2018b]. These contributions have in common the goal of enriching the scalar reward signal to encourage the network to pick up additional structure in the environment, either with the downstream objective of using this information later for policy improvement, or as a more stable learning signal for the network. Additional work has analyzed the geometry [Bellemare et al., 2019] and stability [Ghosh and Bellemare, 2020] of the learned features in deep RL agents, along with their linear algebraic properties [Kumar et al., 2021, Gogianu et al., 2021].

A second challenge in the application of deep neural networks to RL arises from the nonstationarity of the learning objective. The challenge of training a network on a sequence of tasks has been studied in great depth in both reinforcement learning [Schaul et al., 2019, Teh et al., 2017, Igl et al., 2021] and supervised learning settings [Sharkey and Sharkey, 1995, Ash and Adams, 2020, Beck et al., 2021]. Of particular interest has been the problem of catastrophic forgetting, with prior work proposing novel training algorithms using regularization [Kirkpatrick et al., 2017, Bengio et al., 2014, Lopez-Paz and Ranzato, 2017] or distillation [Schwarz et al., 2018, Silver and Mercer, 2002, Li and Hoiem, 2017] approaches. Benjamin et al. [2019], apply a function-space regularization approach, but require saving input-output pairs into a memory bank. However, in reinforcement learning the converse problem must also be considered: forward-interference, whereby networks may see deteriorating ability to solve later tasks encountered during training. Methods which involve re-initializing a new network have seen particular success at reducing interference

between tasks in deep reinforcement learning [Igl et al., 2021, Teh et al., 2017, Rusu et al., 2016, Fedus et al., 2019b].

Interference

Even in the absence of an explicit abstraction-learning objective, generalization between states in deep RL will naturally arise from the network's inductive bias. In many problem settings, such as in large observation spaces or procedurally generated environments, some degree of generalization is necessary in order to obtain good performance at test time [Kirk et al., 2021]. However, when an update to the predicted value of one state exerts excessive influence on the network's output at other states, this form of generalization (which we will also refer to as *interference*) can lead to instability [Jiang et al., 2021b, Van Hasselt et al., 2018]. The introduction of instability as a result of interference between states is not unique to deep RL; even in the case of *linear* function approximation, off-policy temporal difference learning is not guaranteed to converge to the optimal value function [Baird, 1993, Tsitsiklis and Van Roy, 1996]. Interference has been identified as a barrier to learning progress in both policy-based and value-based problems [Schaul et al., 2019, Fedus et al., 2019b].

There are two principal approaches by which we will quantify this notion of interference. To a first-order approximation, interference is equivalent to *gradient alignment*. To formalize this concept, we let f be a function which takes as input an observation $\mathbf{x}$ and a set of parameters θ (for example, f might be the loss function of a neural network, or one dimension of its output). We define gradient alignment $\mathrm{I}_\nabla(\mathbf{x}, \mathbf{y}; \theta)$ (where ∇ indicates that we are studying the *gradient* of the loss) with respect to some loss function and network architecture parameterized by θ and taking inputs $\mathbf{x}$ and $\mathbf{y}$ as follows.

$$\mathrm{I}_\nabla(\mathbf{x}, \mathbf{y}; \theta) \stackrel{\text{def}}{=} \langle \nabla_\theta f(\mathbf{x}; \theta), \nabla_\theta f(\mathbf{y}; \theta) \rangle \tag{2.13}$$

We may instead measure the effect of a gradient step computed on data point $(\mathbf{x}_1, y_1)$ on the loss of another data point $\mathbf{x}_2 \neq \mathbf{x}_1$ with corresponding target y_2. Letting the

targets y be left implicit we obtain the following, where f is some function as before and ℓ is the optimization objective (which may or may not be equal to f). We let g_ℓ be the update computed by some optimizer which may or may not be equal to the gradient $\nabla_\theta \ell(\mathbf{x}_1, y_1; \theta)$, and which may also depend on the optimizer state η.

$$\mathrm{I}_\Delta(\mathbf{x}, \mathbf{y}; \theta) \stackrel{\text{def}}{=} f(\mathbf{y}; \theta) - f(\mathbf{y}, \theta') \text{ where } \theta' = \theta + \alpha g_\ell(\mathbf{x}; \theta, \eta) \tag{2.14}$$

Interference has been studied under a variety of names. Fort et al. [2019] refer to the gradient alignment between data points (a measure related to I_∇ as 'stiffness', while He and Su [2020] call a notion that more closely resembles I_Δ 'elasticity'.

A number of approaches which specifically reduce the effect of a gradient update for state s on the target $V(s')$ have been shown to improve the stability and robustness of off-policy algorithms [Ghiassian et al., 2020, Lo and Ghiassian, 2019]. Prior works have also endeavoured to rigorously define and analyze interference in deep RL [Liu et al., 2020b,a, Bengio et al., 2020], and to study its role in the stability of offline algorithms [Kumar et al., 2022]. Similarly, some recent methods [Shao et al., 2022, Pohlen et al., 2018] include an explicit penalty which discourages gradient updates from affecting the target values used in TD updates, though this is incidental to the principal contribution of these works. Tuning the degree of interference exhibited by the function approximator to maximize performance thus remains an open problem.

2.3.2 Multiple environments

Most standard formulations of generalization in RL do not concern generalization to unseen states within the training environment, but rather generalization to novel environments, MDPs with different observations, transition dynamics, and reward functions than those seen during training [Kirk et al., 2021]. This generalization problem in deep RL can be decomposed into two categories: generalization of the function approximator to novel observations from an MDP which follows the same underlying dynamics as the training environment, and generalization to novel tasks, where some combination of the reward function, observations, and transition

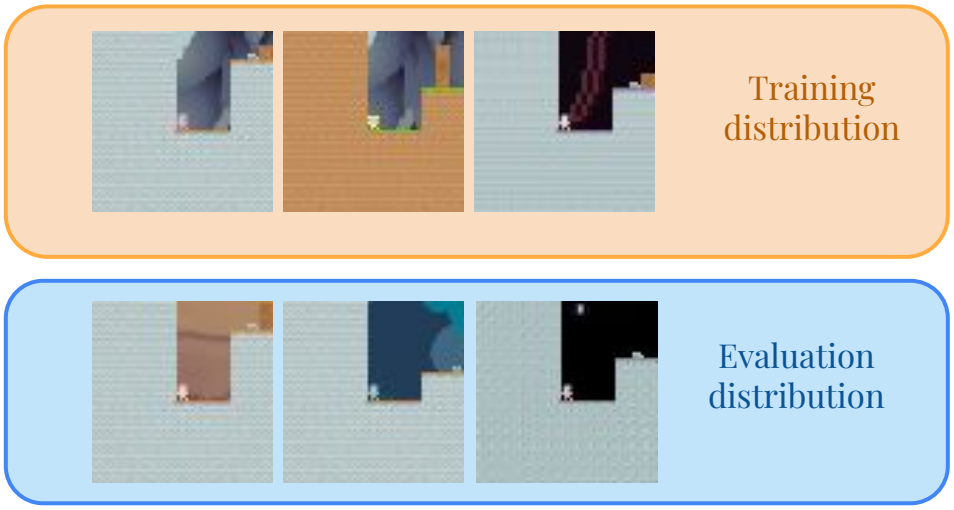


Figure 2.2: Visualization of multi-environment generalization framework with coinrun levels sampled from the ProcGen benchmark [Cobbe et al., 2020].

dynamics are allowed to vary. Many techniques that benefit one of these categories also benefit the other [Packer et al., 2018]. Farebrother et al. [2018] argue, for instance, that an agent which has learned a policy which generalizes well will be robust to changes in the difficulty of a game in addition to more superficial changes in the observation space.

Formalism

We will be concerned with the generalization gap incurred by a policy trained on a set of training environments $\mathcal{E}_{\text{train}}$ when deployed to a novel *test* environment.

$$\mathbb{E}_{\mathcal{E}_{\text{train}},\pi}[\sum_{t=0}^{\infty}\gamma^t R(x_t,a_t)] - \mathbb{E}_{\mathcal{E}_{\text{test}},\pi}[\sum_{t=0}^{\infty}\gamma^t R(x_t,a_t)] \tag{2.15}$$

Typically $\mathcal{E}_{\text{test}}$ will be assumed to share some structure with $\mathcal{E}_{\text{train}}$, but the nature of this shared structure may vary between problem settings. In large observation spaces, $\mathcal{E}_{\text{test}}$ may be equal to $\mathcal{E}_{\text{train}}$ with a different initial state distribution, while for multi-environment problems $\mathcal{E}_{\text{train}}$ may be homomorphic to $\mathcal{E}_{\text{test}}$ [Zhang et al., 2020a]. In the worst case, the evaluation environments may be chosen adversarially so as to make generalization difficult; we will not consider this setting, instead focusing predominantly on test environments that share some characteristics with the training environment.

Procedurally generated environments [Cobbe et al., 2020, Küttler et al., 2020, Samvelyan et al., 2021] enable us to sample an almost-unlimited number of evaluation environments which are visually diverse but which share important structural characteristics. A visualization of frames from different procedurally-generated levels of the *CoinRun* game is given in Figure 2.2. Being able to generate an unlimited set of environments brings us closer to the setting of supervised learning, where training and evaluation data points are sampled from the same distribution, by allowing us to sample training and evaluation *environments* from the same distribution. This regime opens interesting research questions concerning the optimal sampling of environments during training so as to maximize the agent's learning progress and generalization performance [Jiang et al., 2021a, Parker-Holder et al., 2022]. However, the number of training environments in these settings is typically orders of magnitude smaller than the number of training data points seen in standard supervised learning benchmarks, making generalization more challenging.

Regularization

Regularization has been widely shown to improve generalization in deep learning [Srivastava et al., 2014, Krogh and Hertz, 1991], and its application to reinforcement learning tasks has a rich history extending back to even before the explosion of the deep RL paradigm [Farahmand, 2011]. More recent works focus specifically on the regularization of neural networks trained on RL tasks [Cobbe et al., 2019, Farebrother et al., 2018], and show that many regularization methods used to improve generalization in supervised deep learning, such as dropout and ℓ_2 regularization, are also applicable to generalization in reinforcement learning. For example, Schrittwieser et al. [2020] use ℓ_2 regularization in their model-based algorithm which attains state-of-the-art results in many game environments. Many recent works have further sought to leverage the benefits of data augmentation to improve generalization and robustness to input perturbations in reinforcement learning agents [Raileanu et al., 2021, Laskin et al., 2020a,b, Hansen and Wang, 2021]. Li and Pathak [2021] use low-frequency learned Fourier features to regularize deep Q-networks towards

smooth functions. Trust region-based methods further leverage a form of explicit regularization on a learned policy to stabilize learning [Schulman et al., 2015].

The stochasticity induced by some regularization methods serves a dual purpose as a tool for exploration. Igl et al. [2019] use dropout masks to induce stochasticity into the exploratory policy of an agent, while Fortunato et al. [2018] apply a Gaussian perturbation to the network parameters. This approach is also leveraged by the Rainbow [Hessel et al., 2018] architecture. Goyal et al. [2019] use an information bottleneck architecture both to improve generalization and to generate exploration strategies.

Transfer learning and meta-RL

In some problem settings, we seek to train an agent that is capable of quickly adapting to novel environments. In the case where we wish to generalize to novel rewards under the same transition dynamics as we saw during training, this can be achieved via the *successor representation* [Dayan, 1993, Barreto et al., 2017]. In deep RL, we approximate the successor representation using *successor features*, which have been shown to improve sample efficiency and generalization in both single-agent RL and multi-agent settings [Machado et al., 2018b, Filos et al., 2021]. Recently, Abdolshah et al. [2021] present an approach to learning successor features which can generalize across different environments.

However, in many cases the novel environment may feature a novels observation space with different transition dynamics from those seen at training. With the advent of powerful function approximation architectures and the development of effective meta-learning algorithms [Finn et al., 2017, Zintgraf et al., 2019], generalization across multiple environments exhibiting less shared structure may also be approached using transfer- and meta-learning techniques. Teh et al. [2017] and Schmitt et al. [2018], for example, use policy distillation to improve transfer learning to new tasks. Other approaches use context-dependent policies [Rakelly et al., 2019, Sodhani et al., 2021], or encourage invariance across training environments [Zhang et al., 2020b].

2.3.3 Summary

The study of generalization in the reinforcement learning literature has historically been restricted to the framework of *state abstractions*, which consider arbitrary aggregations of states rather than explicit inductive biases on some input space [Li et al., 2006]. Some attempts have been made to define generalization with respect to input observations in single-environment RL, but these definitions run into limitations due to the nonstationary nature of RL, wherein improving prediction accuracy does not necessarily entail improved performance [Bengio et al., 2020, Liu et al., 2020b]. One contribution of this thesis will be to present explicit definitions of interference and learning capacity in deep RL that are less dependent on the chaotic relationship between performance and prediction error, which will be used to study the evolution of learned representations in deep RL agents. We will explore both the benefits, e.g. improved robustness to input distribution shifts, and the drawbacks, e.g. decreased ability to adapt to new reward signals, of generalization between inputs. This exploration will result in algorithmic tools to accelerate learning and improve generalization in deep RL agents. While recent analysis has studied interference between different regions of the state space as a source of instability and a cause of performance plateaus, this line of work largely ignores the beneficial effects of generalization.

In contrast, the benefits of generalization are widely acknowledged in the context of multi-environment RL, where an agent is evaluated on previously-unseen environments at test time. However, the current state of the art in this regime requires prohibitive sample complexity in order to generalize well outside of their training set [Cobbe et al., 2019], largely due to the weak assumptions placed on the generative process from which environments are sampled. While some approaches from the supervised learning literature, such as weight decay, dropout, and data augmentation, have been shown to improve generalization in deep RL, these approaches do not have as great an effect as in supervised learning [Raileanu et al., 2021]. Orthogonal approaches from the meta- and transfer-learning literature have seen more success as they allow for fine-tuning on the test set, though these methods

encounter similar sample complexity issues when applied to zero-shot generalization. In Chapter 8 we will consider a characterization of families of environments in which generalization from a handful of training environments is theoretically possible, and present algorithms for identifying or learning state abstractions for which an optimal policy will generalize zero-shot to any novel environment from this family.

3

The role of invariance in generalization

Contents

3.1 Introduction 44
3.2 Background 46
3.3 Invariance and optimization dynamics 50
3.4 Invariance and generalization bounds 58
3.5 Conclusions 64

3.1 Introduction

The first object of interest in our study of learning dynamics and generalization will be *invariance*. Invariance arises naturally as a part of the data generating process in many real-world datasets, such as point clouds (invariant to permutation), speech recognition (invariant to the average pitch of the speaker's voice), natural language (different sentences may be semantically equivalent), and objects in natural images (often rotation- and translation-invariant). In many instances, an invariance can be precisely characterized in terms of the action of a specified group, meaning that the input distribution is partitioned into equivalence classes, for example all rotations of an image or all permutations of elements in a set. This structure yields straightforward desiderata on the generalization properties of a model trained to fit

such a data-generating distribution: learning about one element of an equivalence class should be sufficient for the model to correctly identify any other element in the equivalence class. In this sense invariance presents an idealized form of *interference*: updates to an invariant model's output on one element of an equivalence class will necessarily have an identical effect across all inputs in this class.

It is not surprising then that models which capture the invariant structure of a given distribution tend to perform better than those which do not on a variety of tasks [Cohen and Welling, 2016, Fawzi et al., 2016, Salamon and Bello, 2017]. Yet while the empirical benefits of invariant models have been widely corroborated, work which seeks to explicitly quantify and understand how invariance can benefit generalization is scarce. Part of this challenge stems from the breadth of approaches by which invariance can be incorporated into a model class. One can build the invariance into the network as a convolution or weight-tying scheme, average network predictions over transformations of the input (feature averaging), or simply train on a dataset augmented with these transformations (data augmentation). Comparison of these different approaches requires different tools, as each incorporates confounding factors into the learning process which are distinct from the effect of the training method on invariance.

This chapter will employ a diverse range of techniques to gain insight into how invariance benefits generalization. The analysis presented here will lay the foundation for our study of more nebulous forms of inductive bias in Chapter 4, and our study of generalization in reinforcement learning in Chapters 7 and 8. We will provide both an empirical and theoretical analysis contrasting neural networks trained to exhibit approximate invariance via data augmentation and networks which exhibit exact invariance via architectural design or feature averaging. Our empirical analysis will study the degree to which data augmentation yields models whose invariances generalize outside of the training set, evaluating the effect of these methods on interference between inputs in the same equivalence class. We further characterize conditions under which data augmentation will provably induce convergence to an invariant model parameterization in Section 3.3.1,

and study the differential effects of data augmentation and feature averaging on interference in Section 3.3.2.

In our theoretical analysis, we will shed light onto the role of invariant structure in PAC-Bayes generalization bounds. Our main contribution will be a ranking of a set of PAC-Bayes bounds on models trained with data augmentation and with feature averaging, resulting in the practical conclusion that feature averaging at test time improves upon data augmentation, which in turn outperforms training on data sampled independently from $P_{\mathcal{D}}$. We find that the differential effects of feature averaging and data augmentation on interference mirror those on the model complexity term in our PAC-Bayes bound. This analogy between model simplicity and interference in the setting of group-theoretic invariances grounds our further study of interference in the learning problems of Chapters 4, 7, and 8 where more nuanced inductive biases must be identified.

3.2 Background

"Invariance" has been used to describe a number of related but distinct phenomena in the machine learning and statistics literature concerning the stability of a function's output under some set of transformations of its input [Zou et al., 2012, Raj et al., 2017, Jaderberg et al., 2015, van der Wilk et al., 2018, Chen et al., 2019]. We focus on invariance under the action of a group $\mathcal{G}$, a setting shared by several prior works [e.g., Cohen and Welling, 2016, Kondor and Trivedi, 2018, Bloem-Reddy and Teh, 2019]. The action of $\mathcal{G}$ on a set $\mathcal{X}$ is a mapping $\alpha : \mathcal{G} \times \mathcal{X} \to \mathcal{X}$ which is compatible with the group operation, i.e. $\alpha(e, x) = x$ and $\alpha(g_1, \alpha(g_2, x)) = \alpha(g_1 g_2, x)$. We will abbreviate this operation as $\alpha(g, x) = gx$ when the form of α is not important. The orbit of any $x \in \mathcal{X}$ is the subset $\mathcal{G}_x$ of $\mathcal{X}$ that can be obtained by applying an element of $\mathcal{G}$ to x, $\mathcal{G}_x = \{gx : g \in \mathcal{G}\}$. We note that the sets $\mathcal{G}_x$ are equivalence classes which partition the underlying set $\mathcal{X}$, and will refer to them as the orbits of $\mathcal{G}$.

For mathematical simplicity, we assume $\mathcal{G}$ to be compact, with (unique) normalized Haar measure denoted by λ.[1] Critically, any finite group is compact

[1] λ is analogous to the uniform distribution on $\mathcal{G}$ [Haar, 1933].

under the discrete topology. For the purposes of this chapter, the reader may therefore substitute 'compact' with 'finite' and 'Haar measure' with 'uniform distribution over $\mathcal{G}$' and preserve the correctness (though not the full generality) of the theoretical results.

We denote a random element of $\mathcal{G}$ by G. A mapping $f : \mathcal{X} \to \mathcal{Y}$ is invariant under $\mathcal{G}$ (or $\mathcal{G}$-invariant) if

$$f(gx) = f(x) \,, \quad \forall\, g \in \mathcal{G},\ x \in \mathcal{X} \,. \tag{3.1}$$

The act of transforming an arbitrary function $f : \mathcal{X} \to \mathbb{R}$ to be $\mathcal{G}$-invariant is known as symmetrization, and can be performed by averaging the output of f over the orbits of $\mathcal{G}$. We define this symmetrization operator $S_{\mathcal{G}}$ as follows:

$$f^{\circ}(x) := S_{\mathcal{G}} f(x) = \mathbb{E}_{G\sim\lambda}[f(Gx)] \,, \quad x \in \mathcal{X} \,. \tag{3.2}$$

While other symmetrization methods exist, such as taking the supremum over the group's orbit, we will always use f° to refer to the *average* of f over the orbit of its input. We consider a typical machine learning scenario, with a training data set $\mathcal{D}^n$ of n observations $(X_i, Y_i)_{i=1}^n \in (\mathcal{X}, \mathcal{Y})^n$ sampled i.i.d. from some (unknown) probability distribution $P_{\mathcal{D}}$. Additionally, $P_{\mathcal{D}}$ *is assumed to be* $\mathcal{G}$*-invariant*, i.e.

$$P_{\mathcal{D}}(gX, Y) = P_{\mathcal{D}}(X, Y) \,, \quad g \in \mathcal{G} \,. \tag{3.3}$$

For example, X may be an image of an animal, Y a label of the animal, and $\mathcal{G}$ the group of two-dimensional rotations.

Given a group action on the space $\mathcal{X}$, we can partition it into disjoint *orbits* of the form $\{g\Phi : g \in \mathcal{G}\}$, where $\Phi \in \mathcal{X}$ denotes an orbit representative. The $\mathcal{G}$-invariance of $P_{\mathcal{D}}$ enables the decomposition of $P_{\mathcal{D}}$ into a distribution over orbits P_Φ along with a conditional distribution $P_{X|\Phi} = \lambda(g)$ where $g\Phi = X$ [see, e.g., Bloem-Reddy and Teh, 2019].

Letting $G \sim \lambda(\mathcal{G})$, this entails $(X, Y) \overset{D}{=} (G\Phi, Y)$ and $P_{\mathcal{D}} = P_\Phi \times P_{X|\Phi} \times P_{Y|X}$. In other words, the distribution $P_{\mathcal{D}}$ can be interpreted as first sampling an equivalence class Φ, then sampling an element X uniformly at random from the set Φ, and then sampling Y based on the conditional distribution $P_{\mathcal{D}}(Y|X)$.

This decomposition will play a key role in our analysis in later sections, as it will allow us to iterate expectations over $P_{\mathcal{D}}$ as

$$\mathbb{E}_{(X,Y)\sim P_{\mathcal{D}}}[f(X,Y)] = \mathbb{E}_{\Phi\sim P_{\Phi}}[\mathbb{E}_{G\sim\lambda(\mathcal{G})}[\mathbb{E}_{Y\sim P_{Y|X}}[f(G\Phi,Y) \mid \Phi, Y] \mid Y]] \,.$$

This will allow us to marginalize over and condition on orbits of $\mathcal{G}$, making it possible to reason about the expected behaviour of a function within and between orbits.

3.2.1 Modes of invariance

When the data-generating distribution and target function are known to be $\mathcal{G}$-invariant, incorporating this invariance into the model class is an intuitive approach to accelerate convergence and improve generalization. This can be done in a variety of ways.

Trained invariance is implemented as data augmentation (DA) [Fawzi et al., 2016, Cubuk et al., 2019]: elements G_{ij} of $\mathcal{G}$ are applied to each observation X_i of the training data, with the label Y_i left unchanged. The result is an augmented dataset $\mathcal{D}^n_{\mathcal{G}} = ((G_{ij}X_i, Y_i)_{j\leq m})_{i\leq n}$ used to minimize the augmented empirical risk

$$\widehat{R}^{\circ}_{\ell}(f, \mathcal{D}^n) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{G\sim\lambda}[\ell(f(GX_i), Y_i)] \tag{3.4}$$

which can be approximated using samples as follows

$$\widehat{R}^{\circ}_{\ell}(f, \mathcal{D}^n) \approx \frac{1}{nm}\sum_{i=1}^{n}\sum_{j=1}^{m} \ell(f(G_{ij}X_i), Y_i) = \widehat{\widehat{R}}^{\circ}_{\ell}(f, \mathcal{D}^n) \,. \tag{3.5}$$

DA is now a standard method in the practitioner's toolkit [Iyyer et al., 2014, Zhou and Troyanskaya, 2015, Salamon and Bello, 2017, Zhao et al., 2018], and recent theoretical work has further established connections to variance reduction methods [Dao et al., 2019]. A major benefit of data augmentation is its versatility: while our study in this chapter focuses on augmentations that can be expressed as the action of a group, data augmentation can also be used to promote other properties of the target function such as smoothness by leveraging a broader class of transformations.

Architectural invariance restricts the function class being learned to contain only invariant functions, typically through either feature averaging (FA) or symmetric function composition. FA symmetrizes the output of an arbitrary neural

network by computing an average over $\mathcal{G}$ at one or more layers, such that the overall network is invariant under the action of $\mathcal{G}$. In practice, averaging is typically done at the penultimate or final layer, resulting in a $\mathcal{G}$-invariant network f°. A network f with D layers is written as the composition of $h_D \circ \cdots \circ h_1$, with the shorthand $h_d^{d'}$ referring to the composition of layers d through d'. The empirical risk of a network with FA at layer d evaluated on $\mathcal{D}^n$ is

$$\widehat{R}_\ell(f^\circ, \mathcal{D}^n) = \frac{1}{n}\sum_{i=1}^{n} \ell\Big(h_d^D \circ \mathbb{E}_{G\sim\lambda}[h_1^{d-1}(GX_i)], Y_i\Big) \,.$$

The development of *invariant* or *equivariant* neural network architectures enables users to avoid the potentially costly computation of this expression for a number of different group invariances. A body of literature of varying degrees of generality has developed to propose new architectures which characterize invariant function classes under a variety of group actions, and to study their properties [Wood and Shawe-Taylor, 1996, Ravanbakhsh et al., 2017, Kondor and Trivedi, 2018, Bloem-Reddy and Teh, 2019, Cohen et al., 2019, Cohen and Welling, 2016].

3.2.2 PAC-Bayes generalizations bounds

As discussed in Section 2.2.1, PAC-Bayes bounds characterize the risk of a randomized prediction rule; the randomization can be interpreted as a posterior distribution over functions Q that may depend on $\mathcal{D}^n$, though this distribution need not correspond to the exact posterior of a Bayesian model. A PAC-Bayes bound on generalization error is typically expressed in terms of a sum of the empirical risk and the Kullback-Leibler (KL) divergence [Kullback and Leibler, 1951] between Q and a fixed prior distribution P, which as mentioned in Chapter 2 can be interpreted as characterizing the *complexity* of functions in the support of P.

We define the risk of a probability distribution Q defined on a function class F in the natural way,

$$R_\ell(Q) = \mathbb{E}_{f\sim Q} R_\ell(f) \quad \text{and analogously} \quad \widehat{R}_\ell(Q) = \mathbb{E}_{f\sim Q}\widehat{R}_\ell(f) \,.$$

PAC-Bayes bounds are highly correlated with generalization performance [Jiang et al., 2020], and in some cases provide non-vacuous results for even highly over-parameterized neural networks [Dziugaite and Roy, 2017, 2018b, Zhou et al., 2019]. The following is a standard bound due to Catoni [2007], which holds for general data generating distributions $P_{\mathcal{D}}$ evaluated on the 0-1 loss.

Theorem 3.2.1 (Catoni [2007])**.** *Let $\mathcal{D}^n$ be sampled i.i.d. from $P_{\mathcal{D}}$, and let ℓ be the binary 0-1 loss. For any prior P and any $\delta \in (0,1)$, with probability $1-\delta$ over samples $\mathcal{D}^n$, for all posteriors Q and for all $\beta > 0$,*

$$R_\ell(Q) \leq \frac{1 - e^{-\beta \widehat{R}_\ell(Q,\mathcal{D}^n) - \frac{1}{n}(\mathrm{KL}(Q \,||\, P) + \log \frac{1}{\delta})}}{1 - e^{-\beta}} \,. \tag{3.6}$$

For bounded loss functions, analogous bounds rely on a sum of the empirical risk and a function of the KL divergence [see, e.g., Dziugaite and Roy, 2017, 2018b]. We state all results only for variations of Catoni's bound (3.6), but analogous findings hold for bounds of the form proposed by McAllester [1999].

3.3 Invariance and optimization dynamics

We begin with an analysis of the manner in which data augmentation and feature averaging influence the optimization dynamics and resulting function output by gradient descent. This analysis will yield two main insights, both concerning the setting of convex losses: first, that while data augmentation can cause gradient descent to converge to a $\mathcal{G}$-invariant solution in certain realizable settings, in general it leads to networks that are only approximately invariant. In fact, these networks can overfit to the training dataset such that their variance over orbits of out-of-distribution data increases over the course of training. Second, we will show that feature averaging, in addition to ensuring $\mathcal{G}$-invariance, reduces the variance of both the loss and its gradients. We will relate this finding to the notion of gradient interference discussed in Section 2.3.1, illustrating the effect of trained and architectural invariance on interference within and between orbits of a group $\mathcal{G}$.

3.3.1 Data augmentation and trained invariance

To begin our study of optimization dynamics and invariance, we lay out some basic properties of the symmetrized risk $\widehat{R}_\ell^\circ$ and its interaction with invariant models. When the loss function is convex, Jensen's inequality can be applied to the augmented risk to compare DA and FA risk estimates. We note that the relevant notion of convexity to this line of argument is with respect to the loss function's first argument, not the parameters of the model. Many (though not all) widely used loss functions are convex in this way (e.g., squared error, cross-entropy). We provide counterexamples highlighting the necessity of this property in Appendix A.2.1.

Proposition 3.3.1. *Let $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ be a loss function that is convex in its first argument. Then for any $f : \mathcal{X} \to \mathcal{Y}$,*

$$\widehat{R}_\ell(f^\circ, \mathcal{D}^n) = \widehat{R}_\ell^\circ(f^\circ, \mathcal{D}^n) \leq \widehat{R}_\ell^\circ(f, \mathcal{D}^n) \ ,$$

and therefore analogous inequalities hold for $\widehat{R}_\ell(Q^\circ, \mathcal{D}^n)$, $R_\ell(f)$, and $R_\ell(Q)$. Furthermore, if $\ell(f(\bullet), \bullet) \in L_2(P_\mathcal{D})$ (i.e., has finite second moment), then

$$\mathrm{Var}_{\mathcal{D}^n \sim P_\mathcal{D}^n}\left[\widehat{R}_\ell(f^\circ, \mathcal{D}^n)\right] \leq \mathrm{Var}_{\mathcal{D}^n \sim P_\mathcal{D}^n}\left[\widehat{R}_\ell^\circ(f, \mathcal{D}^n)\right] \ .$$

The proof of this result can be found in Appendix A.1; it follows from Jensen's inequality and the invariance of the data-generating distribution to augmentations, and states that the beneficial effect of an invariant function on the expected risk stems from its reduced variance over orbits. This observation can be flipped to characterize the detrimental effect of incorporating an invariance in the model which is not present in the data: by mapping all elements of an orbit to the same output, the model's expected risk can be lower bounded by a function of the average variance of the true labels over orbits.

In principle, symmetrizing the risk function via data augmentation should regularize the training procedure towards invariant functions; however, in most settings it is not possible to guarantee that the learned function will be exactly $\mathcal{G}$-invariant, particularly on inputs that differ from those seen during training.

Neural networks trained with data augmentation will in general exhibit low but nonzero variance over equivalence classes, and their failure to generalize this low variance to out-of-distribution is illustrated in Figure 3.1. One notable setting in which it *is* possible to obtain such guarantees is linear function approximation with groups admitting a linear representation over the input space. To state the result, we define a data-generating distribution $P_{\mathcal{D}}$ supported on the set $\mathcal{X} \times \mathcal{Y}$, where $\mathcal{Y} \subseteq \mathbb{R}$ and $\mathcal{X} \subseteq V \simeq \mathbb{R}^d$ is a subset of a d-dimensional vector space over $\mathbb{R}$, V, with dual vector space V^*. We assume that $\mathcal{X}$ spans V. Furthermore, let $\mathcal{G}$ admit a linear representation [Serre, 1977], $\rho : \mathcal{G} \to GL(V)$, with corresponding dual $\rho_g^* = \rho_{g^{-1}}^\top$. Finally, we assume that $P_{\mathcal{D}}$ is invariant to the action of $\mathcal{G}$ and that $\mathbb{E}[y|\mathbf{x}] = \langle \mathbf{w}^*, \mathbf{x} \rangle$ for some unique $\mathbf{w}^*$.

Proposition 3.3.2. *Let $P_{\mathcal{D}}$ and $\mathbf{w}^*$ be defined as above, and let $\ell(\mathbf{w}) = \mathbb{E}_{\mathbf{x},y\sim P_{\mathcal{D}}}[(\langle \mathbf{w}, \mathbf{x} \rangle - y)^2]$. Then the (global) minimizer $\mathbf{w}^*$ of ℓ satisfies $\rho_g^* \mathbf{w}^* = \mathbf{w}^*$ for λ-almost all $g \in \mathcal{G}$. In particular, the function $f(\mathbf{x}) = \langle \mathbf{w}^*, \mathbf{x} \rangle$ is $\mathcal{G}$-invariant.*

In brief, this result shows that in the case of linear models, data augmentation with respect to certain types of symmetry can induce *exact* invariance in the learned function. The proof of Proposition 3.3.2 can be found in Appendix A.1. It follows straightforwardly from the condition that the group act linearly on the input space along with the linearity of the function $f(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$, as in this case the symmetrization of the parameters $\mathbf{w}$ is equivalent to the symmetrization of the function, so by Jensen's inequality any minimizer of ℓ must be invariant to the group action. Under suitable step-size conditions (e.g. the Robbins–Monro conditions) this proposition implies that SGD will converge to an invariant set of weights. Thus, to learn a predictor that exhibits the desired invariance on the entire dataset, it is sufficient to train with SGD on augmented data with a convex loss.

Once non-linearity is introduced into the model architecture, Proposition 3.3.2 will no longer hold as the action of the group on the inputs can no longer be translated to an action on the network's parameters. In the worst case data augmentation may simply result in the function approximator memorizing an invariance on its

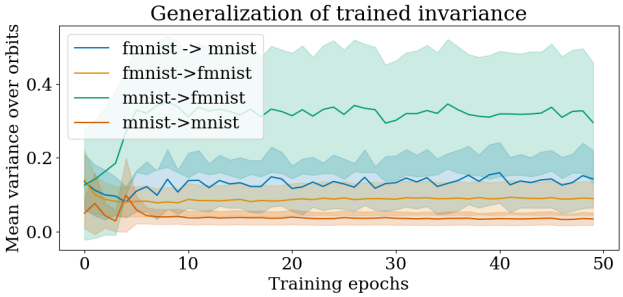


Figure 3.1: Variance of predictions with respect to 90 degree rotations of an input over the course of training. Evaluations performed on either the same dataset as used during training, or on out-of-distribution images. Labels are of the form: training set → evaluation set, where `fmnist` refers to FashionMNIST and `mnist` to MNIST.

training set. The example depicted in Figure 3.1 demonstrates such a failure: the learned function appears to capture the target invariance on the training data, but, having not learned the appropriate symmetry in weight space, fails to generalize this invariance to out-of-distribution data. We measure the generalization of a learned invariance by computing the variance of the network outputs over orbits of data drawn from a dataset distinct from the training distribution. In this case, we train fully connected neural networks using DA on one of two related datasets: MNIST and FashionMNIST (28×28 pixel black and white images of handwritten digits and clothing categories respectively), each augmented by rotations of multiples of 90 degrees. We then evaluate the variance of the outputs over orbits (rotations by 0, 90, 180, and 270 degrees) in the test set. Finally, we evaluate the two networks on orbits in the complementary dataset. In one sense the results are predictable: networks attain lower variance over orbits of within-distribution as compared to out-of-distribution data points. Somewhat surprising is the discrepancy in out-of-distribution generalization between models trained on MNIST and FashionMNIST. As the FashionMNIST inputs are more visually diverse than the MNIST inputs, one explanation of this phenomenon is that the learned invariances are capable of interpolating between training points but struggle to extrapolate, for some loose notion of interpolation and extrapolation.

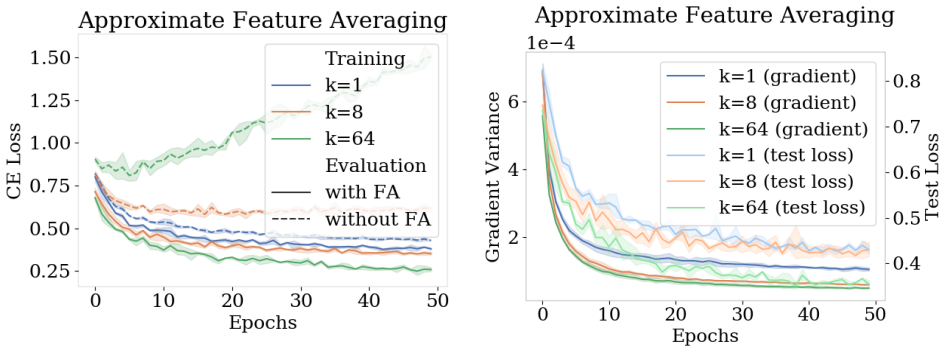


Figure 3.2: Measurements over the course of training a CNN using different DA and FA approaches. *Left*: Models are trained with approximate FA using k rotations sampled from the range $\{1, \ldots, 360\}$, and then evaluated with and without approximate FA. Note that for $k = 1$, the approximate versions of DA and FA are equivalent. *Right*: Average per-epoch gradient variance and test loss in the same setting.

Key insight.
Data augmentation can lead to invariant parameter configurations in restricted settings, but is prone to memorize invariances on its training distribution, leading to out-of-distribution generalization failures.

3.3.2 Symmetrization

Unlike the approximate invariance induced by data augmentation, architectural invariance will trivially generalize to any data-generating distribution. Architectural invariance may also improve computational efficiency by reducing the number of parameters needed to fit the training data.

Feature averaging and Jensen's inequality

Jensen's inequality suggests that incorporating FA at evaluation time will improve performance on convex losses. It does not, however, directly yield insights into whether it is worthwhile to also deploy FA during training. Feature averaging, even when performed approximately, changes the mapping between network parameters and functions, and it is not immediately obvious whether this will benefit the optimization dynamics of the learning process. One potential detractor of training with FA is that the network may not learn parameters which can attain a low loss in its absence.

Hypothesis 3.3.1: models trained with feature averaging will be more dependent on FA to attain a low loss compared to those trained with data augmentation.

To test Hypothesis 3.3.1, we train a series of convolutional neural networks on an augmentation of the FashionMNIST dataset. The labels of this dataset are invariant to rotations: put simply, there is no way of rotating a shoe such that it can be mistaken for a t-shirt. We augment FashionMNIST via the group $\mathcal{G}$ of rotations in the set $\{1°, \ldots, 360°\}$, and perform approximate FA with k samples, where $k << |\mathcal{G}|$. We see in Figure 3.2 that feature averaging profoundly changes the optimization dynamics of the network. The left plot demonstrates that the model trained with FA becomes increasingly dependent on feature averaging to obtain a low loss: the loss of the network's output in the absence of FA increases during training, particularly for larger k (better approximations of FA); it is only when averaging over orbits that the network attains the lowest loss. The right hand side confirms that the models trained with feature averaging exhibit a variance reduction in their gradients and a reduced test loss. The models trained with approximate FA thus converge to regions of parameter space which are distinct from those learned by data augmentation, and which do not yield accurate predictions in the absence of FA.

Invariance and interference

A hallmark of memorization occurs when the network's predictions on its training data match the target function, but its gradients are orthogonal, suggesting it is not able to use information from one input to improve its predictions on others. We therefore study the alignment of gradients, i.e. *interference*, within and between orbits of $\mathcal{G}$; this will amount to generalizing the quantity I_∇ described in Equation 2.13 to apply to *sets* of states. Colinearity of gradients is a much stronger notion of invariance than predictive variance over an orbit. While the latter can easily be set to zero by simple memorization, gradient alignment requires that the network correctly generalize changes in its predictions from one input to others in its orbit. The gradients of a network evaluated on inputs from the same

equivalence class under FA will by necessity be colinear; we now investigate whether this property also holds in models trained with DA.

We study gradient colinearity by computing the rank of the matrix of gradients of inputs computed on a minibatch $\mathbf{X}$ of independently sampled data points, along with the rank of the matrix of gradients of inputs in a batch which contains the full orbit of a subset of these inputs $\mathbf{X}_\Phi$. We subsample $\mathbf{X}$ when generating $\mathbf{X}_\Phi$ to ensure that both batches contain the same number of inputs. Concretely, given a batch of data $\mathbf{X}, \mathbf{y}$ of size n and parameters $\theta \in \mathbb{R}^d$ we are interested in the matrix $\mathbf{I}_\nabla(\mathbf{X}) \in \mathbb{R}^{n \times d}$ defined as follows

$$[\mathbf{I}_\nabla(\mathbf{X})]_{i,j} \stackrel{\text{def}}{=} \nabla_{\theta_j} \ell(f_\theta(\mathbf{X}_i), \mathbf{y}_i) \tag{3.7}$$

where ℓ is a loss function and f_θ is the function computed by the neural network with parameters θ. By considering $\mathbf{X}$ sampled independently and $\mathbf{X}_\Phi$ containing entire equivalence classes, we can distinguish between gradient interference over the whole dataset, and gradient interference specifically along an orbit. An invariant neural network architecture will produce the same gradient for every input from a given equivalence class, and thus the rank of $\mathbf{I}_\nabla(\mathbf{X}_\Phi)$ will be upper bounded by the number of distinct equivalence classes. A non-invariant model might in the worst case assign orthogonal gradients to elements of the same equivalence class; however, such a model could also assign colinear gradients to inputs from different equivalence classes, thus attaining a low rank on $\mathbf{X}_\Phi$ despite exhibiting no invariance. In this case, the network would also attain low rank on $\mathbf{X}$, giving us a means of identifying whether a low gradient rank is due to interference across or within orbits.

In order to be robust to small perturbations of the matrix $\mathbf{I}_\nabla$, we use the numerical rank [Golub et al., 1976] and srank [Kumar et al., 2021], which count the number of (resp. normalized by the maximum) singular values of a matrix that lie above some threshold ϵ. In our evaluations we set $\epsilon = 0.01$ in accordance with Kumar et al. [2021]. Note that the singular values of $\mathbf{I}_\nabla$ are equal to the eigenvalues of the matrix whose entries are given by the dot product $\mathrm{I}_\nabla(\mathbf{x}_i, \mathbf{x}_j)$. We plot these values over the course of training in Figure 3.3. Our training data consists of the

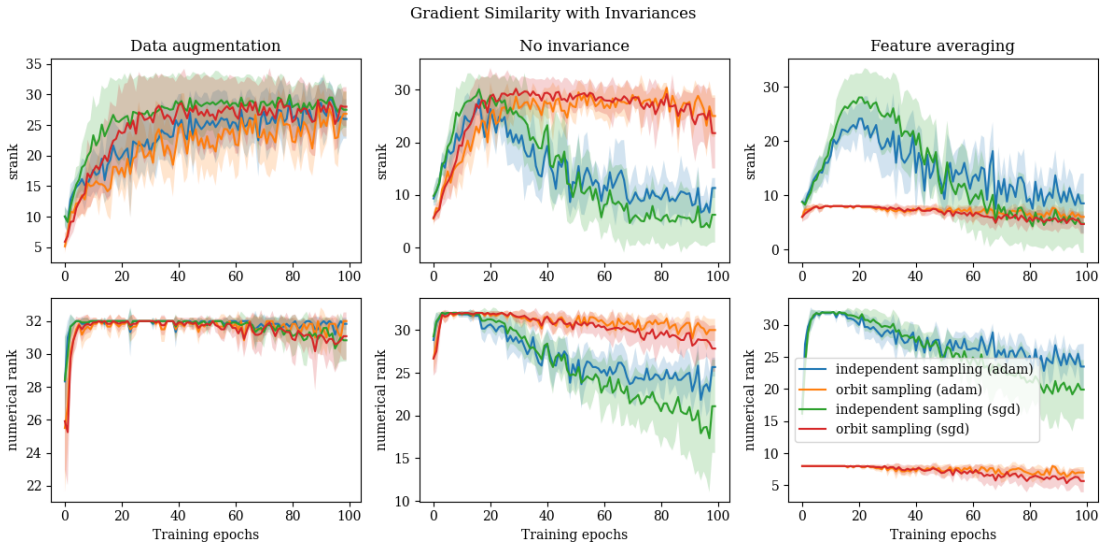


Figure 3.3: Comparing the ranks of gradients computed on minibatches consisting of 32 independent random input samples, or 8 random samples from the training set augmented with all 90 degree rotations (resulting in 32 non-independent samples). While the gradients for inputs from the same orbit are colinear for the feature averaging networks, as evidenced by the low rank of the gradients of augmented minibatches compared to those on independently sampled inputs, the networks without this architectural invariance produce linearly independent gradients. Indeed, the model trained only on non-rotated images exhibits greater independence on rotations than it does on independent random samples from the training set.

CIFAR-10 dataset augmented by rotations of 90 degrees and the evaluations are run on both stochastic gradient descent (SGD) and adaptive (Adam [Kingma and Ba, 2015]) optimizers. For computational tractability, we use a batch size of 32 and subsample parameters in the network to compute the gradient dot products. Despite the batch size being orders of magnitude lower than the number of parameters, we still find that the gradient matrices in most cases do not exhibit full row rank.

We see in Figure 3.3 that networks trained with FA naturally have colinear gradients over orbits, while networks trained without FA do not. In particular, the network trained with DA does not appear to distinguish between elements from the same equivalence class and elements from different classes, at least as far as gradient alignment is concerned. Intriguingly, the network trained without data augmentation exhibits *higher* gradient rank on the augmented minibatches than on the independently sampled data, exhibiting the opposite trend of the network trained with FA. In combination with Proposition 3.3.2, this suggests that while

data augmentation is capable of learning invariant functions over its training set, it does not capture the same degree of invariance in its gradient structure as FA.

3.4 Invariance and generalization bounds

Our previous analysis showed that data augmentation alone is not sufficient to induce invariances in a model that generalize to arbitrary inputs. Instead, models trained with data augmentation have the potential to learn an invariance only on the training distribution, in a sense memorizing the invariant structure. We now explore whether this phenomenon is mirrored in generalization bounds. We will focus in particular on PAC-Bayes bounds, which can be decomposed into a *data fit* and a *model complexity* term. The previous section showed that symmetrization improves the data fit (when this term is given by a convex loss function) of a function class on invariant data-generating distributions. In this section, we will show that invariance can also reduce a notion of model complexity, and combine these two findings to show that invariance leads to lower PAC-Bayes generalization bounds. These results will be complementary to our previous analysis of the training dynamics of neural networks trained with data augmentation and feature averaging.

3.4.1 Validity of the augmented risk

The PAC-Bayes bound in Theorem 3.2.1 holds for a binary classification loss. However, the average of the 0-1 loss over an orbit will lie in the continuum, i.e. $\mathbb{E}_{G\sim\lambda}[\ell(f(GX),Y)] \in [0,1]$. Monte Carlo approximations of $\mathbb{E}_{G\sim\lambda}[\ell(f(GX),Y)]$ also violate the assumptions of Theorem 3.2.1 because the augmented data set is not identically distributed (note that sampling two elements of the same orbit in the augmented dataset will be guaranteed, whereas this event will have vanishing probability under $P_{\mathcal{D}}$). Encouragingly, the following theorem shows that neither of these issues prevents us from leveraging the augmented risk in PAC-Bayes bounds.

Theorem 3.4.1. *Assume that $P_{\mathcal{D}}$ is $\mathcal{G}$-invariant. Then Theorem 3.2.1 holds with $\widehat{R}^{\circ}_{\ell}(Q,\mathcal{D}^n)$ as in (3.4) substituted for $\widehat{R}_{\ell}(Q,\mathcal{D}^n)$.*

The proof of Theorem 3.4.1 can be found in Appendix A.1. The key step in obtaining this result is to define a convex function which measures the difference between $\widehat{R}_\ell(f)$ and $R_\ell(f)$ (which can be translated to a bound on the generalization gap), and then apply Jensen's inequality to show that substituting $\widehat{R}^\circ_\ell$ for $\widehat{R}_\ell$ in this expression will yield a lower bound on this difference, which can then be used to obtain the bound of Theorem 3.2.1.

3.4.2 Invariance and model simplicity

In the case of architectural invariance, particularly via FA, we can go beyond the validity of the PAC-Bayes bound and show a provable reduction in the model complexity term, resulting in a symmetrization gap in the PAC-Bayes bound. Given a function class F, we consider the set F°, denoting the class of $\mathcal{G}$-invariant functions obtained by symmetrizing the functions belonging to F. Symmetrization is a surjective map from F to F° by its definition. However, symmetrization is not necessarily injective, meaning that F° is in a sense smaller than F, and so a distribution over this set should be *simpler*. It is then intuitive that symmetrization should also reduce the model complexity term appearing in a generalization bound. This section will characterize these intuitions rigorously. In order to do so, we must first construct an analogous symmetrization operator on probability measures as we saw for functions in (3.2). This is fortunately straightforward: for any probability measure P on F, the induced probability measure on F° is the image of P under $S_\mathcal{G}$, $P^\circ = P \circ S_\mathcal{G}^{-1}$. Intuitively, incorporating invariances into an architecture should reduce its complexity by forcing P and Q to agree on this property; this intuition is quantified in the following lemma.

Lemma 3.4.2. *Suppose that $(E_i, \mathcal{E}_i)$, $i = 1, 2$, are two measurable spaces, the second of which is standard, μ and ν are two probability measures on $(E_1, \mathcal{E}_1)$, and $\psi : (E_1, \mathcal{E}_1) \to (E_2, \mathcal{E}_2)$ is a measurable map. Then*

$$\mathrm{KL}(\mu \circ \psi^{-1} \,||\, \nu \circ \psi^{-1}) \leq \mathrm{KL}(\mu \,||\, \nu) \,. \tag{3.8}$$

Furthermore, if $\mu \ll \nu$ *with density* m*, then* $\mu \circ \psi^{-1} \ll \nu \circ \psi^{-1}$ *with density* m_ψ*, and the* ψ*-gap is*

$$\begin{aligned}\Delta_\psi(\mu \mid\mid \nu) &:= \mathrm{KL}(\mu \mid\mid \nu) - \mathrm{KL}(\mu \circ \psi^{-1} \mid\mid \nu \circ \psi^{-1}) \\ &= \int_{E_1} \mu(dx) \log \frac{m(x)}{(m_\psi \circ \psi)(x)} \,. \end{aligned} \tag{3.9}$$

We provide a sketch here for intuition, and defer the full proof of Lemma 3.4.2 to Appendix A.1. The lemma leverages the chain rule of the relative entropy to decompose the KL divergence between two distributions into a component which marginalizes over equivalence classes of ψ in the space E_1, and one which conditions on them. We will use the pushforward measure $\mu \circ \psi^{-1}$ (and anlogously $\nu \circ \psi^{-1}$), defined over sets in $\mathcal{E}_2$ as $(\mu \circ \psi^{-1})(A) = \mu(\psi^{-1}(A))$, to 'marginalize' over equivalence classes. Concretely, for probability measures P and Q on the measurable space $\mathcal{E}_1 \times \mathcal{E}_2$, under mild assumptions to guarantee that the following terms are well-defined, we have

$$\mathrm{KL}(P(X_1, X_2) \mid\mid Q(X_1, X_2)) = \mathrm{KL}(P(X_1) \mid\mid Q(X_1)) + \mathrm{KL}(P(X_2|X_1) \mid\mid Q(X_2|X_1)) \,. \tag{3.10}$$

By constructing the joint space $(\mathcal{E}_1, \mathcal{E}_2)$ to correspond to (F°, F), we can design a probability measure on this space with the property that the left hand side of the equality in (3.10) will be equal to $\mathrm{KL}(\mu \mid\mid \nu)$, while the right hand side will consist of the sum $\mathrm{KL}(\mu \circ \psi^{-1} \mid\mid \nu \circ \psi^{-1})$, the KL divergence of the pushforward measures $\mu \circ \psi^{-1}$ and $\nu \circ \psi^{-1}$, and $\Delta_\psi(\mu \ll \nu)$, which measures the degree to which the density of μ varies over equivalence classes of ψ. If the density m is constant wherever it is not zero, then this gap will be zero.

The symmetrization gap. Lemma 3.4.2 simply states that when we compress the input space E_1 via a mapping ψ, we analogously compress probability distributions over this space and obtain a subsequent reduction in the KL divergence of the corresponding pushforward measures. The precise gap in the KL divergence has the closed form (3.9) which measures the degree to which the measure μ varies over the equivalence classes induced by ψ. This indicates that symmetrization can reduce the KL divergence term in the PAC-Bayes bound (3.6), as we show in the following theorem.

Theorem 3.4.3. *Let $\mathcal{X}$ be a compact metric space and $\mathcal{Y}$ a Polish space, $\mathcal{G}$ a group acting measurably on $\mathcal{X}$, and $F = C(\mathcal{X}, \mathcal{Y})$ the class of continuous functions $\mathcal{X} \to \mathcal{Y}$.*[2] *Let Q and P be probability measures on F such that $Q \ll P$ with density q, and $Q^\circ \ll P^\circ$ (density q°) their images under $S_\mathcal{G}$ on F°. Then*

$$\mathrm{KL}(Q^\circ \mid\mid P^\circ) \leq \mathrm{KL}(Q \mid\mid P) \, .$$

Furthermore, the symmetrization gap is

$$\Delta^\circ(Q \mid\mid P) = \mathbb{E}_{f \sim Q}\left[\log \frac{q(f)}{q^\circ(S_\mathcal{G} f)}\right] . \tag{3.11}$$

Proof of Theorem 3.4.3. The proof of this result follows straightforwardly from Lemma 3.4.2; we need only show that the conditions of the lemma hold. To see this, we note that for $\mathcal{X}$ a compact metric space and $\mathcal{Y}$ a Polish space, the space $F = C(\mathcal{X}, \mathcal{Y})$ of continuous functions $f : \mathcal{X} \to \mathcal{Y}$ is a Polish space, and therefore it (along with its Borel σ-algebra $\mathcal{B}(C(\mathcal{X}, \mathcal{Y}))$) is a standard Borel space. For a group $\mathcal{G}$ acting measurably on $\mathcal{X}$, the symmetrization operator $S_\mathcal{G} : F \to F^\circ$ is measurable, and the product space $(F \times F^\circ, \mathcal{B}(F) \otimes \mathcal{B}(F^\circ))$ is a standard Borel space. Thus, the conditions of Lemma 3.4.2 are satisfied and the result follows. □

In practice, computing the expectation over orbits required by exact feature averaging may be intractable. Instead, one may sample a set of k transformations with which to average the function output. While this will not output the exact expectation, it still takes advantage of a simplification of the function space via Lemma 3.4.2, by aggregating functions that have some probability of being mapped to the same approximately averaged function. To formalize the idea, let $g^k = \{g_1, g_2, \ldots, g_k\}$ be a set of k elements of $\mathcal{G}$, and G^k a random realization sampled i.i.d. from λ. Let $S_{g^k} f(x) = k^{-1} \sum_{j \leq k} f(g_j x)$ denote the approximate symmetrization of f by g^k. Finally, let $\widehat{Q^\circ_{g^k}} = Q \circ S_{g^k}^{-1}$ denote the image of a distribution Q on F under S_{g^k}. The following result is a consequence of the fact that Lemma 3.4.2 is true for every g^k, and that for $g^{k+1} = g^k \cup \{g_{k+1}\}$, $S_{g^{k+1}} f(x) = f(g_{k+1} x) + \frac{k}{k+1} S_{g^k} f(x)$.

[2] The result can hold for other function classes F; the key requirement is that conditioning is properly defined in F and F°.

Proposition 3.4.4. *Assume the conditions of Lemma 3.4.3. Let $G_s = G_1, G_2, \ldots$ be a sequence of elements sampled i.i.d. from λ. Then with probability one over G_s,*

$$\begin{aligned}\mathrm{KL}(Q \,||\, P) &\geq \mathrm{KL}(\widehat{Q^\circ_{G^1}} \,||\, \widehat{P^\circ_{G^1}}) \geq \cdots \\ &\geq \mathrm{KL}(\widehat{Q^\circ_{G^k}} \,||\, \widehat{P^\circ_{G^k}}) \geq \cdots \\ &\geq \mathrm{KL}(Q^\circ \,||\, P^\circ)\,.\end{aligned}$$

As the results of Section 3.3.1 suggest, the interplay between SGD and symmetrization remains an open question. However, Proposition 3.4.4 makes it clear that at least at evaluation time, feature averaging is preferred. Indeed, an intriguing feature of this result is that the model simplicity improvement in the PAC-Bayes bound holds even if FA was not applied during training, meaning that the final bound benefits from both the variance reduction on the empirical risk and the ψ-gap provided by symmetrization.

Key insight.
Feature averaging, even when only performed approximately, reduces the model complexity term in PAC-Bayes generalization bounds. The *symmetrization gap* quantifies the degree to which feature averaging simplifies the resulting function class.

Though its exact computation will be intractable for FA in neural networks, the symmetrization gap in the theoretical bounds corresponds to real improvements of generalization, as shown by our empirical analysis in Section 3.4.4. However, the question of causality remains open. Do invariant models generalize better *because* of the reduced KL penalty? Or do they generalize better and obtain better generalization bounds for some other reason? Understanding the answer to this type of question has been the focus of the long line of literature outlined in Section 2.2.2. It is likely that the underlying mechanism relating the two also appears in our analysis from Section 3.3.1: the symmetrization gap arises because an invariant function's output on one element of an equivalence class uniquely determines its output over the whole class. This makes the class of functions computed by a given network architecture smaller, which is reflected in the reduced KL divergence. A

similar notion of simplicity when FA is used during training can be seen in the gradient structure illustrated by Figure 3.3, where an update to one element of an equivalence class automatically generalizes to all other elements in that class.

3.4.3 Ordering of PAC-Bayes bounds

Combining the previous results yields an ordering of the PAC-Bayes generalization upper bounds. Let B_0 be the upper bound on the right-hand side of (3.6), with B_{DA} and B_{FA} corresponding to the upper bounds for DA (using the augmented empirical risk $\widehat{R}_\ell^\circ(Q, \mathcal{D}^n)$) and FA (using $\mathrm{KL}(Q^\circ \,||\, P^\circ)$), respectively. Finally, let $B_{\mathrm{DA}\,*}$ denote the computationally intractable bound for DA given in Appendix A.2.2.

Theorem 3.4.5. *Assume the conditions of Theorem 3.2.1, and also that $P_{\mathcal{D}}$ is $\mathcal{G}$-invariant. Then $B_{\mathrm{FA}} \leq B_{\mathrm{DA}\,*} \leq B_{\mathrm{DA}} = B_0$.*

The proof of Theorem 3.4.5 follows directly from our previous results. Importantly, this ordering on upper bounds does not imply a strict ordering on generalization error. It is nonetheless encouraging, as prior work has shown that many design choices which reduce PAC-Bayes risk bounds result in a reduced generalization gap. We investigate whether this finding holds for feature averaging in Section 3.4.4.

3.4.4 PAC-Bayes bounds for invariant architectures

We conclude with a demonstration of the effect of invariance on PAC-Bayes bounds for neural networks. We use the ModelNet10 dataset, which consists of LiDAR point cloud data for 10 classes of household objects. This dataset exhibits permutation invariance: the LiDAR reading is stored as a sequence of points defined by $\{x, y, z\}$ coordinates, and the order in which the points are listed is irrelevant to the class. We consider three different architectures: a PointNet-like architecture [Qi et al., 2017], which is invariant to permutations; a partitioned version of the PointNet architecture which is invariant to subgroups of the permutation group (details in Appendix A.3); and a fully connected model where the invariant pooling operation

Table 3.1: Generalization performance for a permutation-invariant point cloud classification task (see text for details).

Network	Train Error	Test Error	KL Divergence	PAC-Bayes Bound
Fully connected	0.002	0.65	24957	1.75
Partial-PointNet	0.172	0.248	1992	0.67
PointNet	0.24	0.245	944	0.533

in the PointNet is replaced by a fully connected layer. The invariance in the network is implemented via a max-pooling layer instead of an averaging layer and so is not a direct application of feature averaging; however, the results of (3.8) still apply due to the injective mapping over functions induced by max-pooling.

We compute the PAC-Bayes bounds following the procedure in Dziugaite and Roy [2017]: we convert a deterministic network to a stochastic network by adding Gaussian noise to the weights, and then train this stochastic model using a differentiable surrogate loss that bounds the true PAC-Bayes bound. After this training procedure converges, we then compute the PAC-Bayes bound. We attain an ordering consistent with our theoretical results: the invariant architecture attains the lowest bound, followed by the partially invariant architecture, and finally followed by the fully connected network. We provide a decomposition of the distinct terms in the bound in Table 3.1. While the fully-connected architecture obtains the lowest loss on the training set, it significantly overfits to this data and we see a large generalization gap which is mirrored in the PAC-Bayes bound. In contrast, the invariant architectures exhibit a reduced PAC-Bayes bound and improved accuracy on the test set, despite obtaining a greater loss on the training set.

3.5 Conclusions

This chapter has posed a simple question: how do different approaches to incorporating invariance into a model influence its generalization performance? To answer this question, we have proposed a novel PAC-Bayes generalization bound which

applies to models trained with or without data augmentation. Our theoretical analysis gives the answer that invariant model classes are simpler, where this change in simplicity can be quantified by the relative 'size' of the invariance relative to the size of the data generating distribution. We have further characterized settings under which gradient descent optimization will converge to solutions which satisfy the invariances exhibited by the data generating distribution.

To gain deeper insight into the interplay between invariance and optimization, this chapter further presented an empirical analysis of the effects of data augmentation and feature averaging on the optimization dynamics of neural networks. For non-convex problems, this analysis reveals that networks develop a notion of approximate invariance that holds on held-out points from the data-generating distribution, but can exhibit increasing variance on the orbits of out-of-distribution data points over the course of training, suggesting that neural networks are able to pick up approximate invariance properties that generalize well within distribution but do not correspond to architectural invariance. Our analysis in this setting has also revealed a variance reduction effect of both data augmentation and feature averaging.

The contributions of this chapter exhibit two principal limitations: we consider only model selection problems concerning the selection of exact invariances, and our generalization bounds are not sufficiently tight to be useful for model selection in most settings of interest. The former allowed us to obtain a precise characterization of the benefit of incorporating symmetries into a model class, but limited the scope of inductive bias which we could consider. The latter is a generic failing of current generalization bounds for neural networks, as outlined in Section 2.2.2. While realizations of these bounds were correlated with generalization in our experiments, the bounds are not intended to directly translate to a model ranking. The following chapter will address both of these issues by performing *marginal likelihood estimation* rather than generalization bound computation, and will be applicable to a broad range of hyperparameter and inductive bias selection problems beyond symmetries.

4

Training speed and model selection

Contents

4.1 Introduction 66
4.2 Background on Bayesian inference 69
4.3 Marginal likelihood estimation with training statistics 72
4.4 Training speed and interference 80
4.5 Conclusions 88

4.1 Introduction

As we saw in the previous chapter, choosing the right inductive bias for a machine learning model, in particular incorporating symmetries exhibited by the data, is critical for good generalization. The problem of *Bayesian model selection* concerns itself with identifying good inductive biases for a given dataset, including but not limited to symmetries. Model selection arises in both Bayesian inference, which seeks to identify a probabilistic model that best captures some distribution $P_{\mathcal{D}}$, and in deep learning in the form of neural architecture search. This chapter will show that computing a measure of *training speed*, the area under a model's learning curve, provides a principled approach to Bayesian model selection. Further, we will show that an analogue of this measure adapted to deep learning, the area

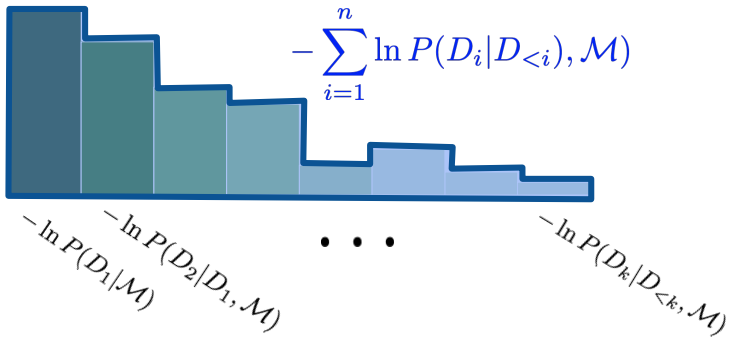


Figure 4.1: Visualization of the relationship between the Bayesian marginal likelihood and training speed. The log predictive likelihood of each data point D_i in a dataset after conditioning the model $\mathcal{M}$ on earlier indices $D_{<i}$ is plotted on the left hand side in green. The shaded blue area on the right corresponds to the area under this curve, i.e. value of the negative log marginal likelihood. This yields the interpretation of the negative log ML as a measure of training speed.

under a network's training curve, produces competitive performance estimators for neural architecture search.

Critical to obtaining these results is our attention to the Bayesian learning paradigm, which is outlined in Section 4.2. In Bayesian inference, the marginal likelihood (ML) provides a principled tool for model selection. In contrast to cross-validation, for which computing gradients is cumbersome, the ML can be conveniently maximized using gradients when its computation is tractable. Unfortunately, computing the marginal likelihood for complex models such as neural networks is typically *in*tractable. Workarounds such as variational inference suffer from expensive optimization of many parameters in the variational distribution and differ significantly from standard training methods for deep neural networks, which optimize a single parameter sample from initialization. A method for estimating the ML that closely follows standard optimization schemes would pave the way for new practical model selection procedures, yet remains an open problem.

The Bayesian model selection perspective is complementary to the study of PAC-Bayes generalization bounds as a tool with which to rank models [Dziugaite et al., 2020], though the two are deeply connected [Germain et al., 2016b]. In parallel, theoretical analysis has hinted at a connection between training speed

and generalization error [Hardt et al., 2016], though the resulting generalization bounds from this analysis are only valid for extremely short training budgets. These results, while providing insight into the stability of stochastic gradient descent, do not present an *explanation* (in the sense of that described in Section 2.2.2) of the empirical observation that models which train faster tend to generalize better. This is because the stability-based generalization bounds of Hardt et al. [2016] implicitly propose a notion of hypothesis class complexity that depends on the number of steps taken in the optimization trajectory, a measure which becomes vacuous after the first training epoch. However, Jiang et al. [2020] show strong empirical performance of some measures of training speed at predicting generalization performance in DNNs, suggesting that this relationship may provide clues towards understanding generalization in deep learning.

The principal contribution of this chapter is to show that Bayesian model selection, training speed, and generalization are in fact deeply connected. To do so, we take a prequential coding perspective on the marginal likelihood, framing the log ML as the sum of predictive log likelihoods of data points, conditioned on preceding data in the dataset. This perspective reveals a family of estimators of the log ML which depend only on predictions sampled from the posterior of an iterative Bayesian updating procedure. We study the proposed estimators in the context of linear models, where we can conclusively analyze their theoretical properties. Leveraging the fact that gradient descent can produce exact posterior samples for linear models [Matthews et al., 2017] and the infinite-width limit of deep neural networks [Matthews et al., 2018, Lee et al., 2018], we show that this estimator can be viewed as the sum of a subset of the model's training losses in an iterative optimization procedure. This immediately yields an interpretation of marginal likelihood estimation as measuring a notion of training speed in linear models, visualized in Figure 4.1.

We demonstrate the utility of this estimator through empirical evaluations on a range of model selection problems, confirming that it can effectively approximate the marginal likelihood of a Bayesian model in a manner useful for model selection.

We go on to investigate whether our theoretical results for linear models may have explanatory power over generalization in deep learning. We construct and justify an analogue of our training speed estimator which can be applied to neural networks trained with stochastic gradient descent, and show that this estimator is predictive of final test accuracy in neural architecture search problems. These findings suggest that studying generalization between points on a training set can be informative of generalization to unseen data, an observation that will inspire our study of interference in reinforcement learning in Chapter 7.

4.2 Background on Bayesian inference

Philosophically, Bayesian model selection addresses the question *which out of this set of models is most likely to have generated my data?* This superficially resembles empirical risk minimization, which seeks to answer the question *which set of (hyper-)parameters minimizes the loss function on my training data?* However, whereas empirical risk minimization is a principle for selecting a single predictor with the goal of minimizing its risk on the data-generating distribution, Bayesian model selection concerns itself with fitting probability distributions over parameters and corresponding functions, without explicit concern for the model's future predictions.

4.2.1 Bayesian modelling

A Bayesian model $\mathcal{M}$ is defined by a prior distribution over parameters θ, $P(\theta|\mathcal{M})$, and a prediction map from parameters θ to a likelihood over the data $\mathcal{D}$, $P(\mathcal{D}|\theta, \mathcal{M})$. Parameter fitting in the Bayesian framework entails finding the posterior distribution $P(\theta|\mathcal{D})$, which yields robust and principled uncertainty estimates. Though exact inference is possible for certain models such as Gaussian processes (GPs) [Rasmussen, 2003], it is intractable for DNNs [Neal, 2012]. Here approximate methods such as variational inference [Blei et al., 2017, Gal and Ghahramani, 2016, Blundell et al., 2015, MacKay, 1992, Graves, 2011, Duvenaud et al., 2016] and Laplace approximations [Daxberger et al., 2021, MacKay, 1998] are used to improve robustness and obtain useful uncertainty estimates.

Variational approximations require optimization over the parameters of the approximate posterior distribution. This optimization over distributions changes the loss landscape, and is significantly slower than the pointwise optimization used in standard DNNs. Pointwise optimization methods inspired by Bayesian posterior sampling can produce similar uncertainty estimates as variational inference, while improving computational efficiency [Welling and Teh, 2011, Mandt et al., 2017, Maddox et al., 2019]. An appealing example of this is ensembling [Lakshminarayanan et al., 2017], which works by training a collection models in the usual pointwise manner, starting from k independently initialized parameter values.

In the case of linear models, marginalization over an ensemble of models trained with gradient descent is exactly equivalent to Bayesian inference, as this sample-then-optimize approach yields exact posterior samples [Matthews et al., 2017, Osband et al., 2018]. He et al. [2020] extend this approach to obtain posterior samples from DNNs in the infinite-width limit.

4.2.2 Model selection

In addition to finding model parameters, Bayesian inference can also perform *model selection* over different inductive biases, which are specified through both model structure (e.g. convolutional vs fully connected Bayesian neural network architectures) and the prior distribution on parameters. The Bayesian approach relies on finding the posterior over models conditioned on the data $P(\mathcal{M}|\mathcal{D})$, which uses the *marginal likelihood* (ML) as its likelihood function:

$$P(\mathcal{D}|\mathcal{M}) = \int_\theta P(\mathcal{D}|\theta)P(\theta|\mathcal{M}_i)d\theta = \mathbb{E}_{P(\theta|\mathcal{M})}[P(\mathcal{D}|\theta)]\,. \tag{4.1}$$

Instead of computing the full posterior, it is common to select the model with the highest marginal likelihood. This is known as type-II maximum likelihood [MacKay, 1992, 2003] and is less prone to overfitting than performing maximum likelihood over the parameters and model combined. This is because the marginal likelihood is able to trade off between model fit and model complexity [Rasmussen and Ghahramani, 2001], while addressing a problem with fewer degrees of freedom

than that of parameter fitting. Maximizing the ML is standard procedure when it is easy to compute. For example, in Gaussian processes it used to set simple model parameters like smoothness [Rasmussen, 2003], while recent work has demonstrated that complex inductive biases in the form of invariances can also be learned [van der Wilk et al., 2018].

For many deep models, computing Equation 4.1 is intractable, and obtaining approximations that are accurate enough for model selection and that scale to complex models is an active area of research [Khan et al., 2019]. In general, variational lower bounds that scale are too loose when applied to DNNs [Blundell et al., 2015] to provide useful estimates for model selection. Deep Gaussian processes provide a case where these bounds do work [Damianou and Lawrence, 2013, Dutordoir et al., 2020], but heavy computational load holds performance several years behind that of deep learning. While ensembling methods provide useful uncertainty estimates and improve the computational efficiency of the variational approach, they have not yet provided a solution for Bayesian model selection.

Further work on generalization in deep learning has examined training dynamics as a tool for understanding how deep neural networks generalize. Many of these works point obliquely at a connection with the marginal likelihood. For example, Arora et al. [2019] obtain generalization bounds for shallow ReLU neural networks that bear striking resemblance to the marginal likelihood of a Gaussian process with kernel given by the neural tangent kernel [Jacot et al., 2018]. The marginal likelihood also appears in other analyses of stochastic gradient descent. Duvenaud et al. [2016] use a single run of stochastic gradient descent to approximate the marginal likelihood of a Bayesian model, and then use tools for model selection to define a criterion for early stopping. Similarly, Smith and Le [2018] use the marginal likelihood to identify the optimal batch size for a learning problem, using the width of the minimum found by stochastic gradient descent as a Laplace approximation for the marginal likelihood of the model.

4.3 Marginal likelihood estimation with training statistics

In this section, we investigate the equivalence between the marginal likelihood (ML) and a notion of training speed in models trained with an exact Bayesian updating procedure. For linear models and infinitely wide neural networks, exact Bayesian updating can be performed using gradient descent optimization. For these cases, we derive an estimator of the marginal likelihood which

1. is related to how quickly a model learns from data,
2. only depends on statistics that can be measured during pointwise gradient-based parameter estimation, and
3. becomes tighter for ensembles consisting of multiple parameter samples.

We also investigate how gradient-based optimization of a linear model combination can implicitly perform approximate Bayesian model selection in Appendix B.1.

4.3.1 Training speed and the marginal likelihood

We begin by developing intuition on the marginal likelihood. Let $\mathcal{D}$ denote a dataset of the form $\mathcal{D} = (\mathcal{D}_i)_{i=1}^n = (x_i, y_i)_{i=1}^n$, and let $\mathcal{D}_{<i} = (\mathcal{D}_j)_{j=1}^{i-1}$ with $\mathcal{D}_{<1} = \emptyset$. We will abbreviate $P(\mathcal{D}^n|\mathcal{M}) \equiv P(\mathcal{D}^n)$ when considering a single model $\mathcal{M}$. We observe that $P(\mathcal{D}^n) = \prod_{i=1}^n P(\mathcal{D}_i^n|\mathcal{D}_{<i}^n)$ to get the following form of the *log* marginal likelihood:

$$\log P(\mathcal{D}) = \log \prod_{i=1}^{n} P(\mathcal{D}_i|\mathcal{D}_{<i}) = \sum_{i=1}^{n} \log P(\mathcal{D}_i|\mathcal{D}_{<i}) = \sum_{i=1}^{n} \log[\mathbb{E}_{P(\theta|\mathcal{D}_{<i})} P(\mathcal{D}_i|\theta)]. \tag{4.2}$$

If we define training speed as the number of data points required by a model to form an accurate posterior, then models which train faster – i.e. whose posteriors assign high likelihood to the data after conditioning on only a few data points – will obtain a higher marginal likelihood. Interpreting the negative log posterior predictive probability $\log P(\mathcal{D}_i|\mathcal{D}_{<i})$ of each data point as a loss function, the

log ML then takes the form of the sum over the losses incurred by each data point during training, i.e. the area under a training curve defined by a Bayesian updating procedure. Crucially, this notion of area under a curve differs from the number of steps required to attain a pre-specified average conditional likelihood or loss. Under the area-under-curve definition of training speed, if model $\mathcal{M}_1$ trains faster than $\mathcal{M}_2$, then for some accuracy ϵ the number of steps required for $\mathcal{M}_1$ to attain average conditional log likelihood ϵ will be lower than for $\mathcal{M}_2$. There is no fixed value of ϵ under which this statement will hold for all possible model pairs; instead, the area under a training curve can be thought of as averaging out many values of ϵ to arrive at a ranking.

Key insight.
The marginal likelihood measures the degree to which a posterior update on each data point increases the likelihood of not-yet-seen data points. It can be computed by taking the area under the loss curve of a particular updating procedure, yielding a notion of *training speed*.

4.3.2 Unbiased estimation of a lower bound

In practice, computing $\log P(\mathcal{D}_i|\mathcal{D}_{<i})$ may be intractable, necessitating approximate methods to estimate the model evidence. In our analysis, we are interested in estimators of $\log P(\mathcal{D})$ computed by drawing k samples of $\theta \sim P(\theta|\mathcal{D}_{<i})$ for each $i = 1, \ldots, n$. We can directly estimate a lower bound $\mathcal{L}(\mathcal{D}) = \sum_{i=1}^{n} \mathbb{E}[\log P(\mathcal{D}_i|\mathcal{D}_{<i})]$ using the log likelihoods of these samples, yielding the estimator

$$\hat{\mathcal{L}}(\mathcal{D}) = \sum_{i=1}^{n} \frac{1}{k} \sum_{j=1}^{k} \log P(\mathcal{D}_i|\theta_j^i). \tag{4.3}$$

This will produce a biased estimate of the log marginal likelihood due to Jensen's inequality. We can get a tighter lower bound by first estimating $\mathbb{E}[\log P(\mathcal{D}_i|\theta)]$ using our posterior samples before applying the logarithm, obtaining

$$\hat{\mathcal{L}}_k(\mathcal{D}) = \sum_{i=1}^{n} \log \frac{1}{k} \sum_{j=1}^{k} P(\mathcal{D}_i|\theta_j^i). \tag{4.4}$$

The two estimators converge when $\mathcal{L}$ is estimated with a single sample and $\hat{\mathcal{L}}_k$ is evaluated at $k = 1$, however we consider them separately in the following theorem due to their disagreement on larger sample sizes.

Proposition 4.3.1. *Both $\hat{\mathcal{L}}$ and $\hat{\mathcal{L}}_k$ as defined in Equation 4.4 are estimators of lower bounds on the log marginal likelihood; that is*

$$\mathbb{E}[\hat{\mathcal{L}}(\mathcal{D})] = \mathcal{L}(\mathcal{D}) \leq \log P(\mathcal{D}) \quad \textit{and} \quad \mathbb{E}[\hat{\mathcal{L}}_k(\mathcal{D})] = \mathcal{L}_k(\mathcal{D}) \leq \log P(\mathcal{D}) \,. \tag{4.5}$$

Further, the bias term in $\mathcal{L}$ can be quantified as follows.

$$\mathcal{L}(\mathcal{D}) = \log P(\mathcal{D}) - \sum_{i=1}^{n} \mathrm{KL}((P(\theta|\mathcal{D}_{<i}) \,||\, P(\theta|\mathcal{D}_{<i+1})) \tag{4.6}$$

The proof of this result follows from a straightforward application of Jensen's inequality and can be found in Appendix B.2. We observe that both lower bound estimators exhibit decreased variance when using multiple posterior samples; however, $\hat{\mathcal{L}}_k$ also exhibits decreasing bias (with respect to the log ML) as k increases; each k defines a distinct lower bound $\mathcal{L}_k = \mathbb{E}[\hat{\mathcal{L}}_k]$ on $\log P(\mathcal{D})$. The gap induced by the lower bound $\mathcal{L}(\mathcal{D})$ is characterized by the information gain each data point provides to the model about the posterior, as given by the KL divergence between the posterior at iteration i and the posterior at iteration $i + 1$. Thus, while $\mathcal{L}$ has a Bayesian interpretation it is arguably more closely aligned with the minimum description length notion of model complexity [Hinton and Van Camp, 1993]. Increasing k in $\mathcal{L}_k$ thus allows us to interpolate between the minimum description length and marginal likelihood maximization principles.

When the posterior predictive distribution of our model is Gaussian, we consider a third approach which, unlike the previous two methods, also applies to noiseless models[1] given in Equation 4.7. We shift focus slightly to the regression setting, where each data point is in the form of an input-label pair (X, y), such that $\mathcal{D}^n = (X_i, y_i)_{i=1}^n$, and $(\theta_j^i)_{j=1}^k$ be k parameter samples from $P(\theta|\mathcal{D}_{<i})$. We assume some uniform distribution over inputs X and a structured conditional distribution consisting of a

[1]Note that the log probability $\log P(\mathcal{D}_i|\theta_j^i)$ will not necessarily be well-defined for noiseless models and so the previous estimators cannot be naively applied.

mapping $f : \Theta \times X \to Y$ which given a set of parameters and an input induces a Gaussian likelihood $P(\cdot|\theta, X) = \mathcal{N}(f(\theta, X), \sigma_N^2)$ for some variance σ_N^2 $P(\cdot|\mathcal{D}_{<i}, X_i)$. The likelihood of a given data point $\mathcal{D}_i = (x_i, y_i)$ under the model thus takes the form

$$P_{\mathcal{M}}(\mathcal{D}_i|\mathcal{D}_{<i}) = \int_\theta \mathcal{N}(y_i|f(\theta, X_i), \sigma_N^2) P(\theta|\mathcal{D}_{<i}) d\theta$$

We can then obtain the following proposition, which characterizes an estimator of a lower bound on $\log \mathcal{P}(\mathcal{D})$.

Proposition 4.3.2. *Let $P(Y_i|\mathcal{D}_{<i}, X_i) = \mathcal{N}(\mu_i, \sigma_i^2)$ for some μ_i, σ_i^2. Define the standard mean and variance estimators $\hat{\mu}_i = \frac{1}{N}\sum_{j=1}^N f(\theta_j^i, x_i)$ and $\hat{\sigma}_i^2 = \frac{1}{N-1}\sum(f(\theta_j^i, x_i) - \hat{\mu})^2$. Then the estimator*

$$\hat{\mathcal{L}}_S(\mathcal{D}) = \sum_{i=1}^n \log P(Y_i|\hat{\mu}_i, \hat{\sigma}_i^2) \tag{4.7}$$

is a lower bound on the log ML: i.e. $\mathbb{E}[\hat{\mathcal{L}}_S(\mathcal{D})] \leq \log P(\mathcal{D})$.

The proof of this result leverages the independence of the sample mean and variance in order to iteratively apply Jensen's inequality to the estimator, and can be found in Appendix B.2. We provide an empirical evaluation of the rankings provided by the different estimators in Section 4.3.3. We find empirically that $\hat{\mathcal{L}}_S$ exhibits the least bias in the presence of limited samples from the posterior, though we emphasize its limitation to Gaussian posteriors; for more general posterior distributions, $\hat{\mathcal{L}}_k$ minimizes bias for large k while still estimating a lower bound.

Lower bounds via gradient descent trajectories

The bounds on the marginal likelihood we introduced in Section 4.3.2 required samples from the sequence of posteriors $P(\theta|\mathcal{D}_{<i})$ as data points were incrementally added. However, such bounds were agnostic to the procedure by which the posterior was sampled. We now draw a connection between Bayesian model selection and gradient descent on linear model classes via a result of Matthews et al. [2017]: ensembles of linear models trained with gradient descent yield samples from the model posterior. In particular, we show that we can use these samples to estimate the log ML using the estimators introduced in Section 4.3.2.

We will consider the Bayesian linear regression problem of modelling data $\mathcal{D} = (x_i, y_i)_{i=1}^n$ assumed to be generated by the process $Y = \theta^\top \Phi(X) + \epsilon \sim \mathcal{N}(0, \sigma_N^2 I)$ for some unknown θ, known σ_N^2, and feature map Φ. Typically, a Gaussian prior is placed on θ; this prior is then updated as data points are seen to obtain a posterior over parameters. In the overparmeterized, noiseless linear regression setting, Matthews et al. [2017] show that the distribution over parameters θ obtained by sampling from the prior on θ_0 and running gradient descent to convergence on the data $\mathcal{D}_{<i}$ is equivalent to sampling from the posterior conditioned on $\mathcal{D}_{<i}$. Osband et al. [2018] extend this result to posteriors which include observation noise $\sigma_N^2 \neq 0$ under the assumption that the targets y_i are themselves noiseless observations.

Algorithm 4.3.1: Marginal likelihood estimation for linear models

Input: A dataset $\mathcal{D} = (x_i, y_i)_{i=1}^n$, parameters $\mu_0, \sigma_0^2, \sigma_N^2$
Result: An estimate of $\mathcal{L}(\mathcal{D})$
$\theta_t \leftarrow \theta_0 \sim \mathcal{N}(\mu_0, \sigma_0^2)$; $\tilde{Y} \leftarrow Y + \epsilon \sim \mathcal{N}(0, \sigma_N^2)$; sumLoss $\leftarrow 0$;
$\ell(\mathcal{D}_{\leq i}, w) \leftarrow \|\tilde{Y}_{\leq i} - \theta^\top X_{\leq i}\|_2^2 + \frac{\sigma_N^2}{\theta_0^2}\|\theta - \theta_0\|_2^2$;
for $\mathcal{D}_i \in \mathcal{D}$ **do**
  sumLoss = sumLoss + $\frac{(\theta_t^\top x_i - y_i)^2}{2\sigma_N^2}$;
  $\theta_t \leftarrow$ GradientDescent$(\ell, \theta_t, \mathcal{D}_{\leq i})$;
end
return sumLoss

We can use this procedure to obtain posterior samples for our estimators by iteratively running sample-then-optimize on the sets $\mathcal{D}_{<i}$. Algorithm 4.3.1 outlines our approach, which executes gradient descent optimization on iterative subsets of the data to obtain the necessary posterior samples for our estimator. We note that the `GradientDescent` subroutine in Algorithm 4.3.1 will in idealized settings output the cluster point of the gradient descent algorithm under decreasing step sizes to give an exact posterior sample; in practice a finite step size may be used and this output will only be an approximate sample from the posterior. Theorem 4.3.3 shows that this procedure yields an unbiased estimate of $\mathcal{L}(\mathcal{D})$ when posterior samples are used to estimate $\mathbb{E}[\log P(\mathcal{D}_i|\theta)]$, and an unbiased estimate of $\mathcal{L}_k(\mathcal{D})$ when an ensemble of k models are trained in parallel to estimate $\mathbb{E}[P(\mathcal{D}_i|\theta)]$.

Theorem 4.3.3. *Let $\mathcal{D} = (X_i, Y_i)_{i=1}^n$ and let $(\theta_j^i)_{i,j=1}^{n,J}$ be generated by the procedure outlined in Algorithm 4.3.1. Then the estimators $\hat{\mathcal{L}}, \hat{\mathcal{L}}_S$, and $\hat{\mathcal{L}}_k$, applied to the collection (θ_j^i), are lower bounds on $\log P(\mathcal{D})$. Further, expressing $-\log P(\mathcal{D}_i|\theta)$ as the ℓ_2 regression loss plus a constant, we then obtain*

$$\log P(\mathcal{D}) \geq \sum_{i=1}^{n} \mathbb{E}_{\theta_i \sim P(\cdot|\mathcal{D}_{<i})}[\log P(\mathcal{D}_i|\theta_i)] = \mathbb{E} \sum_{i=1}^{n} -\ell_2(\mathcal{D}_i, \theta_i) + c = \mathcal{L}(\mathcal{D}) \tag{4.8}$$

We highlight that Theorem 4.3.3 precisely characterizes the lower bound on the marginal likelihood as a sum of training losses based on the regression loss $\ell_2(\mathcal{D}_i, \theta_i)$ when the likelihood $P(\mathcal{D}_i|\theta)$ is a Gaussian.

Infinite-width neural networks

Beyond linear models, our estimators can further perform model selection in the infinite-width limit of neural networks. Using the optimization procedure described by He et al. [2020], we can obtain an exact posterior sample from a GP with kernel equal to the NTK. The iterative training procedure described in Algorithm 4.3.1 will thus yield a lower bound on the marginal likelihood of this GP using sampled losses from the optimization trajectory of the neural network. We evaluate this bound in Section 4.3.3, and formalize this argument in the following corollary.

Corollary 4.3.4. *Let $\mathcal{D}^n$ be a dataset indexed by our standard notation. Let f_0 be sampled from an infinitely wide neural network architecture $\mathcal{F}$ under some initialization distribution, and let f_∞^i be the limiting solution under the training dynamics defined by He et al. [2020] applied to the initialization f_0 and using data $\mathcal{D}_{<i}^n$. Let K_∞ denote the neural tangent kernel for $\mathcal{F}$, and $\mathcal{M} = \mathrm{GP}(\mathbf{0}, K_\infty)$ the induced Gaussian Process. Then $f_\infty^i \sim P(f|\mathcal{D}_{<i}^n, \mathcal{M})$, and in the limit of infinite training time, the iterative sample-then-optimize procedure yields an unbiased estimate of $\mathcal{L}(\mathcal{D}^n|\mathcal{M})$. Letting ℓ_2 denote the scaled squared ℓ_2 regression loss and c be a constant, we obtain as a direct corollary of Theorem 4.3.3*

$$\log P(\mathcal{D}) \geq \mathbb{E}_{f_\infty^i \sim P(\cdot|\mathcal{D}_{<i})}[\log P(\mathcal{D}_i|f_\infty^i)] = \mathbb{E} \sum_{i=1}^{n} -\ell_2(\mathcal{D}_i, f_\infty^i) + c = \mathcal{L}(\mathcal{D})\ . \tag{4.9}$$

It is natural to ask if such a Bayesian interpretation of the sum over training losses can be extended to non-linear models trained with stochastic gradient descent. Although SGD lacks the exact posterior sampling interpretation of our algorithm, we conjecture a similar underlying mechanism connecting the sum over training losses and generalization. Just as the marginal likelihood measures how well model updates based on previous data points generalize to a new unseen data point, the sum of training losses in a stochastic gradient descent trajectory measures how well parameter updates based on one minibatch generalize to the rest of the training data. If the update generalizes well, we expect to see a sharper decrease in the training loss, i.e. for the model to train more quickly and exhibit a lower sum over training losses. This intuition can be related to the notion of 'stiffness' proposed by Fort et al. [2019]. We provide empirical evidence supporting our hypothesis in Sections 4.4 and 4.4.2; for now, we focus on the utility of this estimator in Bayesian model selection.

4.3.3 Empirical evaluation

Section 4.3 focused on two key ideas: that online training statistics can be used in an estimator of a Bayesian model's marginal likelihood (or a lower bound thereof), and that gradient-based optimization can produce the samples needed for this estimation problem. We further conjectured that similar phenomena may also hold for deep neural networks. We now illustrate these ideas in a range of settings. Section 4.3.3 provides confirmation and quantification of our results for linear models, the model class for which we have theoretical guarantees, while Section 4.4.2 provides preliminary empirical confirmation that the mechanisms at work in linear models also appear in DNNs.

While we have shown that our estimators correspond to lower bounds on the marginal likelihood, in order to be useful for model selection also we need the rankings given by the estimators to agree with those assigned by the marginal likelihood. We first evaluate the relative rankings given by the true marginal likelihood with those given by our estimators on a simple feature selection task, consisting of 15 informative features $\phi_1, \ldots, \phi_{15}$ and 15 features containing random

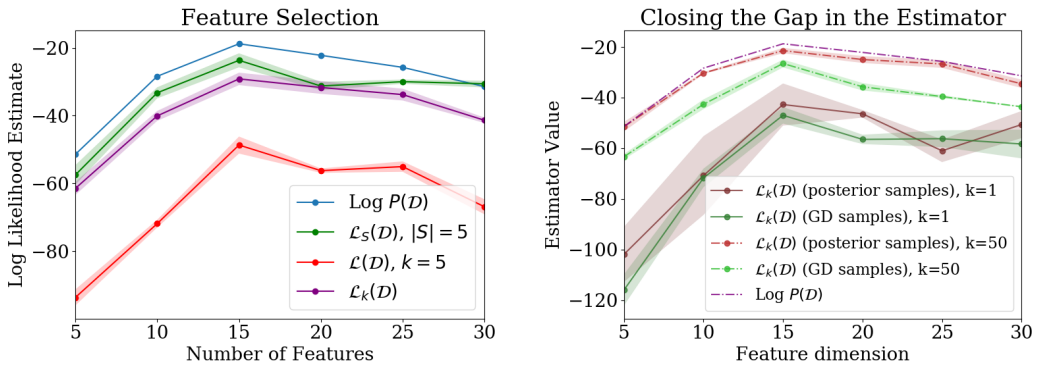


Figure 4.2: Left: ranking according to $\log P(\mathcal{D})$, $\mathcal{L}(\mathcal{D})$ with exact posterior samples, and $\mathcal{L}(\mathcal{D})$ computed on samples generated by gradient descent. Right: gap between true marginal likelihood and $\mathcal{L}_k(\mathcal{D})$ estimator shrinks as a function of k for both exact and gradient descent-generated samples.

noise $\phi_{16}, \ldots, \phi_{30}$. Each model $\mathcal{M}_i$ uses features $\phi_1, \ldots, \phi_i$ in its regression objective and ignores the rest. Full experiment details, along with a description of two additional model selection problems evaluated in Appendix B.1, can be found in Appendix B.3.1. Naturally, the optimal model is $\mathcal{M}_{15}$, which uses all informative features and no extraneous ones. We compare $\mathcal{L}_S$, $\mathcal{L}$ and $\mathcal{L}_k$ to see whether each can identify the optimal model. We first observe that all methods agree on the optimal model: this is a consistent finding across all of the model selection tasks we consider. While all methods lower bound the log marginal likelihood, $\mathcal{L}_k(\mathcal{D})$ and $\mathcal{L}_S(\mathcal{D})$ exhibit a reduced gap compared to the naive lower bound. In the rightmost plot of Figure 4.2, we further quantify the reduction in the bias of the estimator $\mathcal{L}_k(\mathcal{D})$ described in Section 4.3.2. We use exact posterior samples (which we denote in the figure simply as posterior samples) and approximate posterior samples generated by the gradient descent procedure outlined in Algorithm 4.3.1 using a fixed step size and thus inducing some approximation error. We find that both sampling procedures exhibit decreasing bias as the number of samples k is increased, with the exact sampling procedure exhibiting a slightly smaller gap than the approximate sampling procedure.

We further illustrate how our estimator of $\mathcal{L}(\mathcal{D})$ can select inductive biases in the infinite-width neural network regime in Figure 4.3. The left hand side of the figure shows the marginal likelihoods of NTK-GP models for a fully-connected

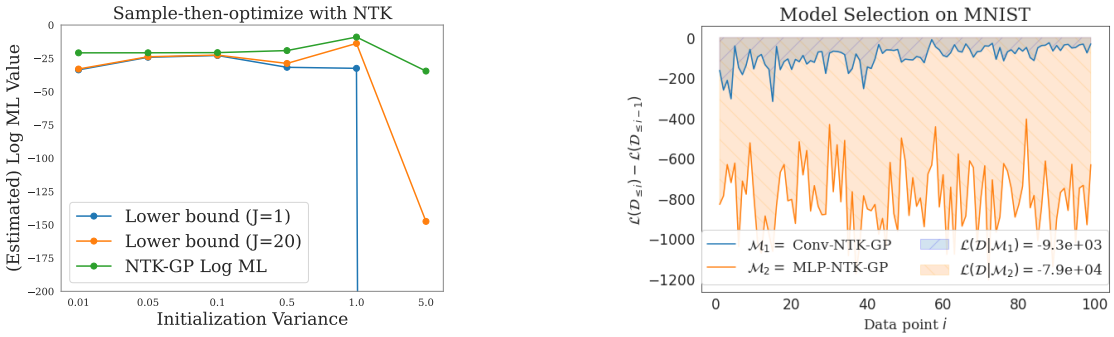


Figure 4.3: Left: approximate lower bound on log ML of NTK model on a prior variance selection task. Right: visualizing the interpretation of $\mathcal{L}(\mathcal{D})$ as the 'area under the curve' of training losses: we plot the relative change in the estimator $\mathcal{L}(\mathcal{D}_{\leq i}) - \mathcal{L}(\mathcal{D}_{<i})$ for convolutional and fully-connected NTK-GP models, and shade their area.

MLP architecture given different prior variances, illustrating that the sample-based estimator $\hat{\mathcal{L}}$ does indeed give a lower bound which becomes more accurate as the number of samples grows. In the right hand side of the figure, we evaluate the relative change in the log ML of a Gaussian Process induced by a fully-connected MLP (MLP-NTK-GP) and a convolutional neural network (Conv-NTK-GP) which regress on the MNIST dataset. The fully-connected model sees a consistent decrease in its log ML with each additional data point added to the dataset, whereas the convolutional model sees the incremental change in its log ML become less negative as more data points are added, as well as a less negative incremental decrease in the log ML at the start of training. This leads to the Conv-NTK-GP having a higher value of $\mathcal{L}(\mathcal{D})$, and a higher log marginal likelihood, than the MLP-NTK-GP. This provides reassurance that the estimators we have proposed behave reasonably in simple architecture selection problems.

4.4 Training speed and interference

Having revealed that statistics from a particular gradient descent training procedure can yield an accurate estimator of the log marginal likelihood, we now turn our attention towards generalizing these findings to the setting of multi-epoch stochastic gradient descent optimization in deterministic function approximators. We begin by relating the posterior predictive likelihoods used in our estimator to the agreement

between the gradients of disjoint minibatches (i.e. interference exhibited by the function approximator) in a dataset. This motivates an analogous estimator for SGD trajectories, which is highly correlated with generalization performance.

4.4.1 Generalization and multi-epoch optimization

We have thus far studied the relationship between training speed and the marginal likelihood in the context of Bayesian models updated iteratively on successive elements of the training data. This has yielded an effective model selection tool based on computing the area under a training curve for specific updating procedures. Translating this approach to the optimization methods used to train deep neural networks faces two major hurdles: first, although it is possible to train neural networks to approximate a Bayesian posterior over weights [Neal, 2012], standard training schemes deal only with a deterministic initialization and so do not directly correspond to probabilistic models. It is possible to argue that many stochastic optimization algorithms sample from an approximate Bayesian posterior [Welling and Teh, 2011, Mandt et al., 2017], but this argument only highlights the second hurdle facing our estimator: most optimization schemes used in practice run over a fixed dataset for several epochs. In contrast, our estimator requires an iterative training procedure whereby each data point is sequentially added to the training set, and we never evaluate the loss on revisited data points.

However, independent of the relationship between gradient descent and Bayesian posterior sampling, the same intuition relating training speed to the marginal likelihood hints at a mechanism by which training speed correlates with generalization error in the gradient descent setting. Recall that the marginal likelihood can be written as the sum

$$\log P(\mathcal{D}) = \sum_{i=1}^{n} \log P(\mathcal{D}_i | \mathcal{D}_{<i}). \tag{4.10}$$

Now consider the term $\log P(\mathcal{D}_i|\mathcal{D}_{<i})$. This quantity characterizes how accurately the model predicts $\mathcal{D}_i$ given that it already knows the values of $\mathcal{D}_{<i}$. The change in the log conditional probability of a data point $\mathcal{D}_{i+1}$ after conditioning on data

point $\mathcal{D}_i$ thus presents a Bayesian analogue of interference as defined in (2.14). We now investigate whether interference in minibatch optimization might provide a means of leveraging the insights described for Bayesian models in a broader class of learning algorithms.

We first establish notation. We denote data $\mathcal{D}^n = \{(x_i, y_i)\}_{i=1}^n$, let ℓ be a loss function, f_θ the function induced by parameters θ, and $R(\theta) = \mathbb{E}_{P_\mathcal{D}}[\ell(f_\theta(x), y)]$. The sequence $(\theta_t)_{t=0}^T$ denotes the series of parameters obtained over a gradient descent trajectory. Recalling that $\log P(\mathcal{D}^n)$ has the interpretation of an area under a training curve, we therefore consider the area under the training curve of a stochastic gradient descent trajectory. For the moment, we will consider minibatches of size one to simplify our analysis; it is straightforward to extend this reasoning to larger minibatches. We first consider a single training epoch, denoting by θ_k the value of the parameters at step k after performing gradient steps $\theta_{t+1} = \theta_t - \alpha \nabla_\theta \ell(x_t, y_t, \theta_t)$. With a slight abuse of notation, we repurpose $\hat{\mathcal{L}}$ to apply to SGD trajectories as follows.

$$\hat{\mathcal{L}}(\mathcal{D}^n) = \sum_{k=0}^{n} \ell(x_k, y_k, \theta_k) \tag{4.11}$$

$$= \sum_{k=0}^{n} [\ell(x_k, y_k, \theta_0) + \sum_{j=1}^{k} (\ell(x_k, y_k, \theta_j) - \ell(x_k, y_k, \theta_{j-1}))] \tag{4.12}$$

Recall that this is precisely our definition of *interference* from Equation 2.14: $\mathrm{I}_\Delta(x_k, x_j)$.

$$= \sum_{k=0}^{n} [\ell(x_k, y_k, \theta_0) + \sum_{j=1}^{k} \mathrm{I}_\Delta(x_j, x_k)] \tag{4.13}$$

$$= \hat{R}_\ell(\mathcal{D}^n; \theta_0) + \sum_{j=1}^{n} \sum_{k>j} \mathrm{I}_\Delta(x_j, x_k) \tag{4.14}$$

The term $\mathrm{I}_\Delta(x_j, x_k)$ measures the effect of the gradient step computed on data point (x_j, y_j) on the loss at the data point (x_k, y_k), thus $\sum_{j<k} \mathrm{I}_\Delta(x_j, x_k)$ measures the cumulative effect of gradient updates computed on earlier data on the loss at point k. This quantity philosophically resembles the $\log P(\mathcal{D}_i | \mathcal{D}_{<i})$ term appearing in the expression of the log marginal likelihood. Indeed, in the first epoch of

training it is an unbiased estimate of the effect of the gradient updates performed up to step k on the expected risk R.

$$\mathbb{E}_{x,y\sim P_{\mathcal{D}}}[\ell(x,y;\theta_k)] = \mathbb{E}_{\mathcal{D}^n\sim P_{\mathcal{D}}}[\ell(x_k,y_k;\theta_k)] \tag{4.15}$$

$$= \mathbb{E}_{\mathcal{D}^n\sim P_{\mathcal{D}}}[\ell(x_k,y_k;\theta_0) + \sum_{j<k} \mathrm{I}_{\Delta}(x_j,x_k)] \tag{4.16}$$

Unbiased estimation of the change in the true risk is only attainable due to the independence of θ_k and x_k, y_k. After the first epoch of training, this property no longer holds as the pair (x_k, y_k) was used to obtain the network's current parameters. Applying similar techniques to those of Hardt et al. [2016] may yield bounds on the bias induced by this dependency, but these tend to be overly pessimistic, resulting in loose upper bounds on the expected risk. Concretely, two problems arise in the case of multiple epochs: first, the loss on the training set will decrease, meaning that the raw change in the loss for any given data point will eventually tend to zero and cease to provide informative updates; second, over the course of many epochs the network may overfit not just its predictions but also its gradient structure to the training set. Of course, neither of these problems is guaranteed to prevent effective performance estimation. In the first case, the losses from earlier in the training trajectory are still likely to be informative, and the low magnitude of the later losses when the network has overfit mean that this later period is unlikely to significantly influence the area under the training curve. In the second case, we note that in general changing the structure of gradients is a much more difficult task than changing predictions (recall the invariance-learning experiments of Figure 3.3), and so we expect the bias induced by these second-order effects to be small. Taking into account the large variance of using a minibatch to estimate the change in loss over the whole dataset, this yields the following practical conjecture.

Hypothesis 4.4.1: the interference between minibatches in the training set $I_{\Delta}(x_i, x_j)$ will exhibit low bias as an estimator of the change in the validation loss relative to its variance due to stochastic optimization.

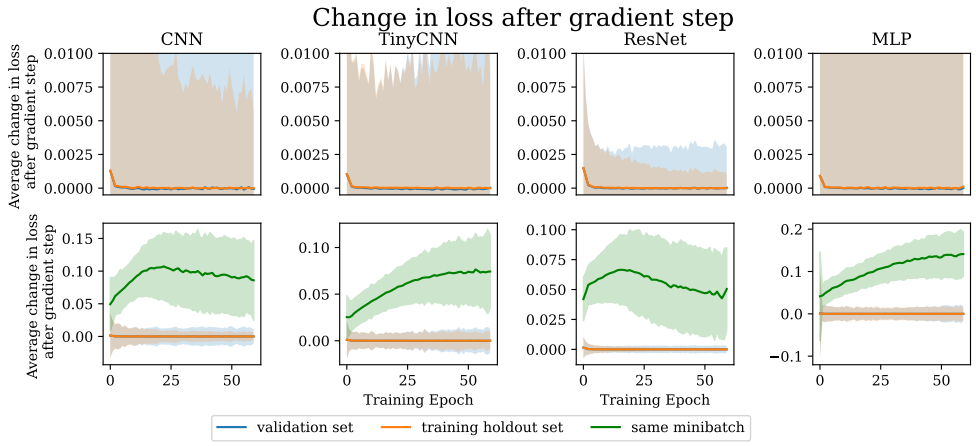


Figure 4.4: Effect of minibatch gradient steps on train (orange), test (blue), and minibatch (green) loss over the course of training a variety of neural network architectures on CIFAR-10. Top and bottom rows plot the same quantities and differ only in the range of the y axis, which shows the average change in the loss before and after a minibatch gradient step in each epoch. Shaded region indicates one standard deviation. Values for the train and test loss are similar, with the difference in means indistinguishable relative to the variance due to minibatch sampling over the epoch, while the change in loss on the minibatch for which the gradient step was taken dwarfs both.

We evaluate Hypothesis 4.4.1 in Figure 4.4, where we plot $\mathrm{I}_\Delta(x_j, x_k)$, for y_k equal to the same minibatch as x_k ('same minibatch'), a larger subset of the training set ('training holdout set'), and a minibatch drawn from the validation set ('validation set'). For the training holdout set and the validation set, we use minibatches of size 5000 to isolate the variance in the estimator due to only the optimizer minibatch sampling. Intriguingly, these results suggest that even after several epochs, $\mathrm{I}_\Delta(x_j, x_k)$ will exhibit low bias as an estimator of $R_\ell(\theta_t) - R_\ell(\theta_{t-1})$ in a range of neural network architectures and optimization schemes. This suggests that smoothing out the variance in the loss estimate induced by stochastic parameter updates will plausibly provide a greater benefit than using an unbiased estimate of performance on the validation set. This is likely to be the case in particular when a large learning rate is used early in training, as is common in practice.

Motivated by these observations, we propose the following generalization measure which directly computes the area under a network's training curve. We refer to it as the SOTL, due to its interpretation as measuring the sum of training losses

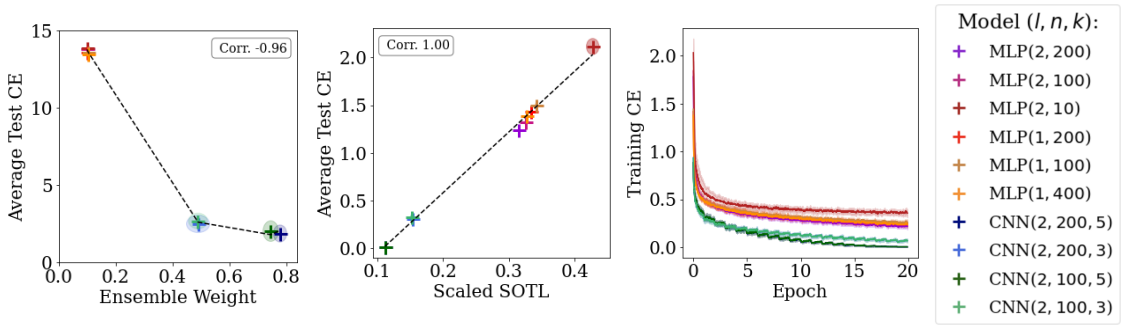


Figure 4.5: Left: ensemble weights versus the test loss for concurrent training. Middle: sum over training losses (SOTL), standardized by the number of training samples, versus test loss for parallel training. Right: training curves for the different models trained in parallel. All results are averaged over 10 runs, and standard deviations are shown by the shaded regions around each observation. The model parameters, given in the parentheses, are the number of layers (l), nodes per layer (n) and kernel size (k), respectively.

from the model's training trajectory.

$$\text{SOTL} = \sum_{t=1}^{T} \left[\frac{1}{B} \sum_{i=1}^{B} \ell \left(f_{\theta_{t,i}}(\mathbf{X}_i), \mathbf{y}_i \right) \right] \tag{4.17}$$

A more detailed study of the SOTL is outside the scope of this thesis, though validation of its utility in neural architecture search can be found in the related work of Ru et al. [2021]. We perform a small-scale proof of concept in this chapter.

Training speed in DNNs

Motivated by the previous discussion, we conjecture that just as the sum of the log posterior likelihoods is useful for Bayesian model selection, the sum of minibatch training losses will be useful to predict generalization error. In this section, we evaluate whether this conjecture holds for a simple convolutional neural network trained on the FashionMNIST dataset. Our results provide preliminary evidence in support of this claim, and suggest that further work investigating this relationship may reveal valuable insights into how and why neural networks generalize. We first evaluate whether the sum over training losses (SOTL) obtained over an SGD trajectory correlates with a model's generalization error, and whether SOTL predicts the weight assigned to a model by a linear ensemble. A discussion of the link between linear ensemble weight and training speed is provided in Appendix B.1; to summarize, we note that (sub-)models with the best sum of training losses will also exhibit the

greatest time-averaged correlation with the target, and so in some sense present the 'best' feature for the linear model combination to use. To evaluate the connection between these three concepts, we train a linear combination of DNNs with stochastic gradient descent and evaluate a) the SOTL of each model's training trajectory, b) the final validation loss of each model, and c) the weight assigned to each model by a linear ensembling layer trained concurrently with the networks.

We observe a strong correlation between the sum of training losses (SOTL) and average test cross-entropy (see Figure 4.5 middle column), validating that the SOTL provides useful model rankings. Further, we find that architectures with lower test error (when trained individually) are given higher weight by the linear ensembling layer – as can be seen from the left plot in Figure 4.5. This finding may have intriguing implications on empirical phenomena such as the lottery ticket hypothesis [Frankle and Carbin, 2019]. Further details of the experiment can be found in Appendix B.3.2. Our results are summarized in Figure 4.5. Though the architecture sets in these illustrative experiments are small, these results are replicated in much larger architecture search spaces by Ru et al. [2021].

4.4.2 Bias and variance in performance estimation

We now investigate *why* the sum of training losses seems to predict generalization despite our departure from the theoretically grounded regime of Bayesian models. We have seen in Figure 4.5 that the SOTL is predictive of generalization performance in simple model selection problems – indeed, further work [Ru et al., 2021] has robustly demonstrated that it is often a *better* predictor of generalization than the early-stopping validation loss. This is quite remarkable as most of the received wisdom in deep learning recommends against using the training set for hyperparameter search in favour of a held-out validation set. We recall Figure 4.4, which revealed that the effect of one minibatch gradient step on the loss of a disjoint minibatch exhibited low bias as an estimator of the change in the expected risk, while the variance of this estimator due to stochasticity in the gradient descent trajectory was itself significant. This suggests that using training speed as a performance

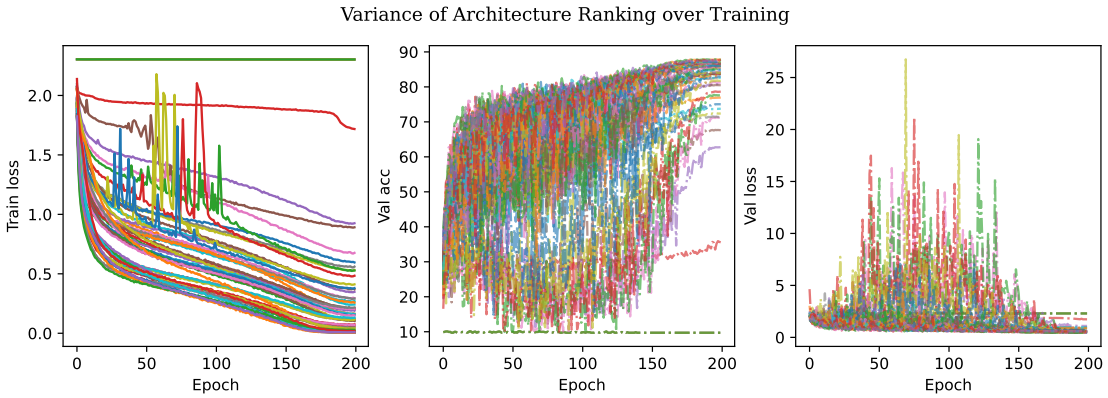


Figure 4.6: The ranking of a randomly-selected subset of models from the NASBench-201 dataset given by training speed estimators is consistent across subsequent epochs, whereas the ranking given by a point estimate of the validation accuracy exhibits high variance between epochs for the same architecture, illuminating the mechanism by which training speed can yield more informative performance estimates for neural architecture search.

estimator may be performing a bias-variance trade-off: training speed introduces a small amount of bias in the performance estimator as the model overfits, but by marginalizing over all of the parameters visited during an epoch or a trajectory, significantly reduces the variance in the performance estimation. Concretely, we obtain the following hypothesis.

Hypothesis 4.4.2: training speed, as measured by the sum of training losses, provides a lower-variance estimate of performance than the validation loss in noisy optimization procedures such as stochastic gradient descent.

We validate this hypothesis in Figure 4.6, where we evaluate a randomly sampled subset of models from the NASBench-201 dataset [Dong and Yang, 2020], the dataset in which variants of the SOTL evaluated by Ru et al. [2021] obtained impressive rank correlation performance, outperforming learning curve extrapolation methods and the early stopping validation loss. We see that the curves corresponding to the validation accuracy and validation loss at each epoch are indistinguishable for most architectures, while the training loss, which is averaged over minibatches sampled during an entire epoch, exhibits much lower variance. This suggests that using the early-stopping validation loss would yield an extremely noisy estimator of final

performance, whereas the SOTL, even if only averaged over the losses of a single epoch late in training, yields a relatively consistent ranking.

Key insight.
The sum of training losses yields an analogous performance estimator to $\mathcal{L}(\mathcal{D})$ which can be applied to neural networks. It measures, among other things, *interference*: the degree to which a gradient update on one minibatch generalizes to the rest of the training set.

4.5 Conclusions

In this chapter, we have proposed a family of estimators of the marginal likelihood which illustrate the connection between training speed and Bayesian model selection. Because gradient descent can produce exact posterior samples in linear models, our result shows that Bayesian model selection can be performed by training a linear model with gradient descent and tracking how quickly it learns. This approach also applies to the infinite-width limit of deep neural networks. The intuition behind this estimator is appealing: a model which can better leverage information from one data point to improve its predictions on other data points will generalize better to new data than one which is not able to do so. We apply similar intuition to propose an analogous performance estimator for deep neural networks, which computes the area under a model's training curve. This estimator obtains high rank correlation with the final generalization error of a neural network, yielding a competitive approach to performance estimation for neural architecture search. We provide evidence that the connections shown in linear models have predictive power towards explaining generalization and training dynamics in DNNs. This empirical analysis highlights the importance of generalization between inputs in the training set, i.e. interference, as a factor influencing both training speed and generalization error.

One limitation of the connection illustrated in this chapter is that it can only be straightforwardly applied to problems of identifying a suitable inductive bias or network architecture. Regularization schemes that influence training dynamics, such as dropout or weight regularization, introduce confounding factors into the training

process under which the relationship between training speed and generalization breaks down. Training speed alone may also be insufficient for problems of optimizer or hyperparameter selection, where these quantities may reduce training speed via their influence on the ability of each optimization step to reduce the loss on the current minibatch, rather than by influencing generalization between minibatches.

While generalization between training inputs will play a key role in our analysis of generalization in reinforcement learning, the results from this chapter cannot immediately be applied to deep RL agents. This is because the estimators we use require a fixed optimization objective to provide a meaningful notion of training speed. In reinforcement learning, both the optimization objective and the input distribution are constantly changing. A deep RL agent that obtains a low training loss quickly may nonetheless achieve sub-optimal return in the environment. In order to understand the role of interference in deep reinforcement learning, it will first be necessary to understand how the dynamics of RL differ from those of the supervised learning setting. It is only with this understanding that we will be able to leverage the insights developed in this chapter and the previous one to predict and improve generalization in deep reinforcement learning.

5 Dynamics of reinforcement learning

Contents

5.1 Introduction 90
5.2 Mathematical framework 92
5.3 Value function & representation dynamics 94
5.4 Auxiliary tasks 100
5.5 Experiments 109
5.6 Conclusions 113

5.1 Introduction

The previous chapters have shown that properties of a network's training dynamics can provide useful insights into its generalization performance. The remainder of this thesis will apply similar analysis into the training of *reinforcement learning* agents. As mentioned in the discussion of TD learning in Section 2.1.2, the dynamics of reinforcement learning agents are less stable than those of supervised learning [Baird, 1993], leading to divergence even for simple linear function approximation schemes. This complexity necessitates a more detailed study of the learning dynamics of neural networks trained with value-based RL algorithms before it is possible to discuss the implications of these dynamics on generalization in full detail. To this end, the current chapter will provide a novel theoretical framework for our

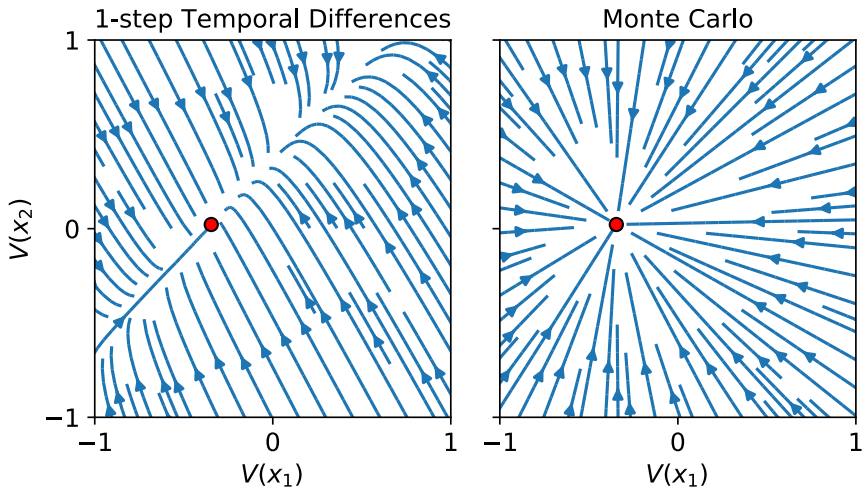


Figure 5.1: An example of qualitatively different value function dynamics for a two-state MDP for 1-step temporal difference learning and Monte Carlo learning, with fixed point V^π in red.

study of representation learning and illustrate its utility by analyzing the effect of various auxiliary tasks on these dynamics; Chapter 6 will draw on insights from this framework to motivate an empirical study of the representations learned by deep RL agents, while Chapter 7 will explore its implications on memorization and overfitting to training observations.

Our theoretical analysis of representation dynamics in RL is motivated in part by the robust improvements given by auxiliary tasks to deep reinforcement learning agents. While supervised learning often benefits from unsupervised pretraining [Radford et al., 2018, Erhan et al., 2010, Lee et al., 2021], fitting auxiliary labels that are only tangentially related to the principal task has not been incorporated into standard learning pipelines. In contrast, training a network to predict not just the value function, but other properties of the environment such as the value of a pixel at the next timestep or expected state occupancies under the current policy, consistently helps RL agents improve their performance [Jaderberg et al., 2017, Mirowski et al., 2017, Lin et al., 2019]. A commonly-held belief is that these benefits are mediated through improved representation learning. This hypothesis naturally raises a number of questions that, broadly speaking, remain open. What makes a good auxiliary task? Can we predict how an auxiliary task will affect

an agent's representation? When should one auxiliary task be used instead of another? More generally, how should this hypothesis about the mechanism of auxiliary tasks itself be tested?

The complex interacting components of large-scale deep reinforcement learning agents make it difficult to extract general insights. In this chapter we aim to shed light on the answers to these questions by distilling the benefits of auxiliary tasks down to their effects on the dynamics of the learned representation. We begin by considering a *learning dynamics* framework for studying the evolution of an agent's predictions and learned representation; see Figure 5.1 for a toy illustration, with full details given in Section 5.2. The central idea behind this framework is that it is not just *what* an agent learns that dictates how its representation is shaped, but also *how* it learns.

This framework provides a model for representation learning in RL. Under this model, even in the case of model-free algorithms, agents can be shown to automatically incorporate the transition structure of the environment into their representations. We characterize the dynamics induced by a number of auxiliary tasks, with particular focus on ensemble predictions and random cumulant functions, and prove convergence of the induced representations to subspaces defined by certain decompositions of the environment's transition operator. These results will play a key role in our analysis in Chapter 7. We then consider the effectiveness of auxiliary tasks in sparse-reward environments, and via the use of the learning dynamics framework, construct a hypothesis as to which auxiliary tasks should be particularly well suited to such environments. We test this conjecture in the Arcade Learning Environment [Bellemare et al., 2013], demonstrating strong performance with random cumulant auxiliary tasks. These findings motivate our study of representation learning in sparse-reward environments in Chapter 6.

5.2 Mathematical framework

This section will present a mathematical framework from which to study the learning dynamics of RL agents, with a particular focus on the dynamics of an

agent's representation. A description of the reinforcement learning setting along with the basic notation used in this chapter can be found in Chapter 2. While we will formulate the representation in terms of the values attained by some feature output, our analysis will focus on the *evolution* of these features over time. In taking this perspective, we lay the groundwork for our later discussion of *capacity* in Chapter 6, where we frame a representation in terms of not just the network's feature outputs under its current parameters, but also the optimization dynamics that these parameters induce.

5.2.1 Features and representations

In many environments, it is impractical to store a value function as a table indexed by states. Even in settings where this is practical, it may not be desirable, as function approximation is necessary for generalization to new observations that the agent may encounter during or after training. Instead, it is typical to parameterize $V \in \mathbb{R}^{\mathcal{X}}$ through a *feature map* $\phi : \mathcal{X} \to \mathbb{R}^K$ and *weight vector* $\mathbf{w} \in \mathbb{R}^K$, leading to a factorization of the form

$$V(x) = \langle \phi(x), \mathbf{w} \rangle .$$

Such a parameterization may be amenable to more efficient learning, for example if ϕ abstracts away unimportant information, allowing for generalization between similar states. Even more concisely, writing $\Phi \in \mathbb{R}^{\mathcal{X} \times K}$ for the matrix with rows $\phi(x)$ yields

$$V = \Phi \mathbf{w} . \tag{5.1}$$

The quantity Φ is often referred to as the agent's *representation* of the environment [Boyan, 1999, Levine et al., 2017, Bertsekas, 2018, Chung et al., 2018, Bellemare et al., 2019, Dabney et al., 2021]. In many small- and medium-scale applications, the representation is fixed ahead of time, and only $\mathbf{w}$ is updated during learning; this is the linear function approximation regime. Many common choices of features relate to various decompositions of mathematical objects associated with the transition operator P^π. In deep reinforcement learning, however, Φ and $\mathbf{w}$ are learned

simultaneously. In this chapter we will consider the representation to be the output of the penultimate layer of the network, meaning that the mapping from representation to value is linear. While it is possible to model Φ as the output of any network layer, the resulting dynamics will not necessarily be analytically tractable as the mapping from features to network outputs will be nonlinear.

5.2.2 Subspace distances

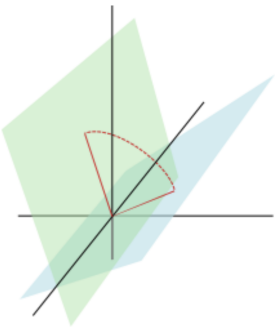


Figure 5.2: Visualizing the Grassmann distance between two 2-dimensional subspaces.

The key results of this chapter will depend on characterizing the convergence of the subspace spanned by a set of feature vectors. To do so, we need a notion of distance between subspaces of $\mathbb{R}^{\mathcal{X}}$. The following definition follows Ye and Lim [2016]. Intuitively, it can be thought of as generalizing the notion of an angle between vectors to one between subspaces.

Definition 5.2.1. *For two K-dimensional subspaces $Y_1, Y_2 \leq \mathbb{R}^{\mathcal{X}}$, the* principal angles $\theta_1, \ldots, \theta_K \in [0, \pi/2]$ *between the subspaces are defined by taking orthonormal matrices $\mathbf{Y}_1 \in \mathbb{R}^{\mathcal{X} \times K}$ and $\mathbf{Y}_2 \in \mathbb{R}^{\mathcal{X} \times K}$ the columns of which span Y_1 and Y_2 respectively, and defining $\theta_k = \cos^{-1}(\sigma_k(\mathbf{Y}_1^\top \mathbf{Y}_2))$, where $\sigma_k(\mathbf{A})$ is the k^{th} singular value of the matrix $\mathbf{A}$. One can check that this definition is independent of the matrices $\mathbf{Y}_1$ and $\mathbf{Y}_2$, depending only on the subspaces Y_1, Y_2 themselves. The* Grassmann distance $d(Y_1, Y_2)$ *between Y_1 and Y_2 is then defined as $\|\theta\|_2 = (\sum_{k=1}^{K} \theta_k^2)^{1/2}$.*

With these definitions in hand, we are now ready to give a precise version of the statement alluded to by the discussion and figure in Section 5.1.

5.3 Value function & representation dynamics

We begin our study of learning dynamics in RL with a discussion of the dynamics followed by a *value function* when Bellman updates are applied. We make some

simplifying assumptions to avoid focusing on technicalities here, and give a discussion of the more general case in Appendix C.4.

5.3.1 Value function dynamics

We consider a continuous-time analogue of one-step temporal difference (TD) learning dynamics. While in practice TD updates are performed discretely, the continuous-time regime lends itself more readily to analysis and has been leveraged frequently in the literature to provide theoretical guarantees on the corresponding discrete-time process [Borkar and Meyn, 2000]. We begin with the following system:

$$\partial_t V_t(x) = \mathbb{E}_\pi[R_0 + \gamma V_t(X_1) | X_0 = x] - V_t(x)\,,$$

for each $x \in \mathcal{X}$, which may also be written

$$\partial_t V_t(x) = R^\pi(x) + \gamma(P^\pi V_t)(x) - V_t(x)\,,$$

or in full matrix notation,

$$\partial_t V_t = -(I - \gamma P^\pi)V_t + R^\pi\,. \tag{5.2}$$

The differential equation in (5.2) is an affine autonomous system, and is straightforwardly solvable.

Lemma 5.3.1. *If $(V_t)_{t\geq 0}$ satisfies Equation (5.2) with initial condition V_0 at time $t = 0$, then we have*

$$V_t = \exp(-t(I - \gamma P^\pi))(V_0 - V^\pi) + V^\pi\,. \tag{5.3}$$

We recover as a straightforward corollary the well-known result that $V_t \to V^\pi$ as $t \to \infty$, since all eigenvalues of $(I - \gamma P^\pi)$ have strictly positive real part.

However, the solution in Equation (5.3) also describes the *trajectory* by which V_t reaches this limiting value. Figure 5.1 provides an illustration of this in a small MDP; the value functions accumulate along a particular affine subspace of $\mathbb{R}^\mathcal{X}$ prior to convergence. This phenomenon can be formalized via the Grassmann distance.

Assumption 1. *P^π is real-diagonalizable, with strictly decreasing eigenvalue sequence $1 = \lambda_1 > \lambda_2 > \cdots > \lambda_{|\mathcal{X}|}$, and corresponding right-eigenvectors $U_1, \ldots, U_{|\mathcal{X}|}$.*

Proposition 5.3.2. *Under Assumption 1, and $(V_t)_{t\geq 0}$ the solution to Equation (5.2), for almost every[1] initial condition V_0, we have*

$$d(\langle V_t - V^\pi \rangle, \langle U_1 \rangle) \to 0\,.$$

Proof sketch. The full proof of this statement and those that follow can be found in Appendix C.2. We provide a sketch as follows: note that we can write the value function trajectory V_t with respect to the eigen-basis $U_1, \ldots, U_{|\mathcal{X}|}$, where $V_0 = \sum_{i=1}^{|\mathcal{X}|} \alpha_0^i U_i$, as $V_t = \sum_{i=1}^{|\mathcal{X}|} \alpha_0^i \exp(-\lambda_i t) U_i = \sum_{i=1}^{|\mathcal{X}|} \alpha_t^i U_i$. This then gives the following form of the error $V_t - V^\pi$

$$V_t - V^\pi = \exp(-t(I - \gamma P^\pi))(V_0 - V^\pi) = \sum_{i=1}^{|\mathcal{X}|} \alpha_i \exp(t(\gamma\lambda_i - 1)) U_i \tag{5.4}$$

$$= \alpha_1 \exp(t(\gamma - 1)) U_1 + o(\exp(t(\gamma\lambda_1 - 1)) \tag{5.5}$$

whenever $\lambda_2 < \lambda_1$. Via a lemma in Appendix C.1, we can show that this condition (5.5) is sufficient to guarantee that $V_t - V^\pi$ converges to U_1 in Grassmann distance. □

We can observe this predicted behaviour in Figure 5.1, where the value function V_t converges to the line $V^\pi + \alpha\mathbb{1}$, i.e. the value function offset by a constant function under temporal difference learning dynamics. Note that $\mathbb{1}$ is an eigenvector of any stochastic matrix, corresponding to eigenvalue 1, the largest eigenvalue (i.e. U_1 for any ergodic P^π will be the constant function). A more general version of this statement can also be given with an ensemble of K value functions, which indicates that yet more information about the environment is contained in the learned collection. The proofs of these results relate to the power method in linear algebra.

[1] In the measure-theoretic sense that the set of excluded initial conditions V_0 has Lebesgue measure 0.

Proposition 5.3.3. *Under Assumption 1, and* $(V_t^{(k)})_{t\geq 0}$ *the solution to Equation (5.2) for each* $k = 1, \ldots, K$*, for almost every initial condition* $(V_0^{(k)})_{k=1}^K$*, we have*

$$d(\langle V_t^{(k)} - V^\pi \mid k \in [K]\rangle, \langle U_{1:K}\rangle) \to 0\,.$$

The proof of this statement is analogous to that of Proposition 5.3.2 and can be found in Appendix C.2. It depends on an eigendecomposition of the transition operator, which allows us to characterize the convergence of the value function along each eigenspace. Because the value function converges more slowly along the principal eigenspaces of P^π, most of the mass of the resulting error term will reside in these spaces as training progresses. The intermediate values of V_t are therefore biased towards the principal subspaces of P^π, and in this sense give us some information about the transition structure of the environment.

Key insight.
Even in an environment with no reward signal at all (in which case $V^\pi = 0$), an agent performing TD learning still picks up information about the transition structure of the environment within its value function.

Due to the importance of the vectors $U_{1:K}$ in this analysis, we introduce the term *eigen-basis functions* (EBFs) to describe them. Intuitively, the eigen-basis functions $U_{1:K}$ provide a basis for the K-dimensional subspace to which V_t converges; we illustrate two demonstrative eigen-basis functions of a random walk on a gridworld in Figure C.2. For any MDP $\mathcal{M}$ and any policy π, we will have that $U_1 = \mathbb{1}$, the constant function. In general, the eigenvectors corresponding to large positive eigenvalues of P^π will be 'smooth' with respect to the transition dynamics, in the sense that they will not vary much in expected value after an application of P^π. This property will be discussed further in Chapter 7.

We observe that a similar analysis, indicating similar behaviour, is possible for related learning algorithms such as n-step temporal difference learning and TD(λ); see Appendix C.5 for further details. In contrast, Monte Carlo learning

dynamics correspond to the differential equation

$$\partial_t V_t = (I - \gamma P^\pi)^{-1} R^\pi - V_t \,,$$

which has the solution

$$V_t = e^{-t}(V_0 - (I - \gamma P^\pi)^{-1} R^\pi) + (I - \gamma P^\pi)^{-1} R^\pi \,.$$

The trajectory associated with this solution simply linearly interpolates between V_0 and V^π, as illustrated in Figure 5.1, and does not pick up any additional information about the environment in the value function as learning proceeds. See Appendix C.5 for further details. This example serves to illustrate that it is not just *what* an agent learns (V^π), but *how* the agent learns (the trajectory V_t) that plays a key, measurable role in what environment information is incorporated in its value function. The notion of information used here is characterized by the subspace in which the learned value function lies; this property naturally relates to linear function approximation in determining the set of functions which can be well-approximated using a given set of features, leading us to consider whether similar subspace convergence properties to those shown for value functions arise in an agent's representation.

5.3.2 Representation dynamics

We now seek to apply similar dynamics analysis to our model of representation learning to better understand the learning dynamics followed by an agent's representation. Recall the parameterization of $V \in \mathbb{R}^{\mathcal{X}}$ from Section 5.2.1, taking the form

$$V = \Phi \mathbf{w} \,,$$

for $\Phi \in \mathbb{R}^{\mathcal{X} \times K}$, $\mathbf{w} \in \mathbb{R}^K$. Central to deep reinforcement learning is the idea that Φ and $\mathbf{w}$ are simultaneously learned from a single RL loss. As in the value function case, we will focus on the dynamics with single-step temporal difference learning; remarks on other learning algorithms are given in Appendix C.5. The dynamics associated with single-step TD learning are given by

$$\partial_t \Phi_t = -\alpha \frac{1}{2} \nabla_{\Phi_t} \| R^\pi + \square[\gamma P^\pi \Phi_t \mathbf{w}_t] - \Phi_t \mathbf{w}_t \|_2^2 \,, \tag{5.6}$$

$$\partial_t \mathbf{w}_t = -\beta \frac{1}{2} \nabla_{\mathbf{w}_t} \| R^\pi + \square[\gamma P^\pi \Phi_t \mathbf{w}_t] - \Phi_t \mathbf{w}_t \|_2^2 \,, \tag{5.7}$$

where $\alpha, \beta \in [0, \infty)$ are learning rates, implying that features and weights may be learned at different rates. Further, $\square[\,\cdot\,]$ denotes a *stop-gradient* on its argument, indicating that we treat the instances of Φ_t and $\mathbf{w}_t$ within the expression as constants when computing derivatives; this reflects the fact that temporal difference learning is a *semi-gradient* method [Sutton and Barto, 2018].

The use of a single loss to learn both the representation and weights corresponds to the approach taken in deep RL, and we will use these dynamics as an idealized model of the dynamics of deep neural networks. While this model ignores some practicalities of deep RL (such as visitation distributions, implicit bias from the function approximation architecture, and stochasticity introduced by minibatch training), it allows us to obtain valuable insights into representation dynamics which, as we will see in Section 5.5 and in the following chapters, accurately predict the behaviour of deep RL agents.

Lemma 5.3.4. *Let Φ_t and $\mathbf{w}_t$ parameterize a value function approximator as defined above. Then*

$$\partial_t \Phi_t = \alpha(R^\pi + \gamma P^\pi \Phi_t \mathbf{w}_t - \Phi_t \mathbf{w}_t)\mathbf{w}_t^\top\,, \tag{5.8}$$

$$\partial_t \mathbf{w}_t = \beta \Phi_t^\top (R^\pi + \gamma P^\pi \Phi_t \mathbf{w}_t - \Phi_t \mathbf{w}_t)\,. \tag{5.9}$$

This joint flow on Φ_t and $\mathbf{w}_t$ leads to much richer behaviour than the flow considered on value functions in the previous section. Without further assumptions, the evolution of the representation Φ_t may be complex, and will not necessarily incorporate environment information as described for the case of value functions in Proposition 5.3.3. In particular, in sparse-reward environments, the agent may learn to predict a near-zero value function by setting the weights $\mathbf{w}_t$ close to zero, which would effectively prevent any further updating of the features Φ_t, ruling out the possibility of a result analogous to Proposition 5.3.3. We will only be able to obtain the upcoming result of Theorem 5.4.1 by restricting the rate at which the weights evolve over the course of training.

Auxiliary task	Dynamics ($r = 0$)	Φ_∞ ($r = 0$)	Φ_∞ ($r \neq 0$)	$\lim_{t\to\infty}\langle\Phi_t - \Phi_\infty\rangle$
Ensemble	$-(I - \gamma P^\pi)\Phi_t$	0	$(I - \gamma P^\pi)^{-1} r\epsilon^\top$	EBFs of P^π
Random cumulants	$-(I - \gamma P^\pi)\Phi_t + Z_\Sigma$	ΨZ_Σ	ΨZ_Σ	EBFs of P^π
Additional policies	$-(I - \gamma P^{\overline{\pi}})\Phi_t$	0	$(I - \gamma P^{\overline{\pi}})^{-1} R^{\overline{\pi}}\epsilon^\top$	EBFs of $P^{\bar{\pi}}$
Multiple γs	$-(I - \overline{\gamma} P^\pi)\Phi_t$	0	$(I - \overline{\gamma} P^\pi)^{-1} R^\pi\epsilon^\top$	EBFs of P^π

Table 5.1: Summary of dynamics and limiting solutions under some common auxiliary tasks in the limit of infinitely-many prediction outputs. For additional policies, $\overline{\pi}$ denotes the average of the finite set of policies $\pi_1, \ldots, \pi_L$ under consideration (L fixed and independent of M), and for multiple discount factors, $\overline{\gamma}$ denotes the average of the discount factors $\gamma_1, \ldots, \gamma_L$ under consideration. We let $\Psi = (I - \gamma P^\pi)^{-1}$. See Section 5.4.2 and Appendix D.1.2 for additional details.

5.4 Auxiliary tasks

Having studied the temporal difference learning dynamics in Equations (5.8) & (5.9), we now examine how auxiliary value-prediction tasks influence the behaviour of the agent's representation during the learning process. As described previously, developing a granular description of the joint learning dynamics of the representation and weights of the learner is a complex task, and so we focus on the limiting case in which the number of auxiliary tasks is large relative to the dimensionality of the representation. We conclude that under certain conditions, representations learned in the many-task limit bear a close connection to the *eigen-basis functions* described in Section 5.3, and also to the *resolvent singular basis functions*, a new decomposition introduced in Section 5.4.2. The reader may find it useful to refer to Appendix C.7 for a more detailed discussion of these decompositions.

5.4.1 Ensemble value prediction

We begin by considering the auxiliary task of *ensemble value prediction* [Osband et al., 2016, Anschel et al., 2017, Agarwal et al., 2020]. Rather than making a single prediction of the value function Q^π, the learner makes $M \in \mathbb{N}$ separate predictions as linear functions of a common representation Φ^M, using M independently initialized weights matrices $\mathbf{w}^m \in \mathbb{R}^K$ $(m = 1, \ldots, M)$. We note that while at initialization $\Phi_0^M \in \mathbb{R}^{\mathcal{X}\times d}$ is independent of M, its dynamics do depend on M through the contribution of the weights. Simultaneous temporal difference learning on all

predictions leads to the following dynamics:

$$\partial_t \Phi_t^M = \alpha \sum_{m=1}^{M} (R^\pi + \gamma P^\pi \Phi_t^M \mathbf{w}_t^m - \Phi_t^M \mathbf{w}_t^m)(\mathbf{w}_t^m)^\top , \tag{5.10}$$

$$\partial_t \mathbf{w}_t^m = \beta (\Phi_t^M)^\top (R^\pi + \gamma P^\pi \Phi_t^M \mathbf{w}_t^m - \Phi_t \mathbf{w}_t^m) . \tag{5.11}$$

The following result characterizes the representation learned by the agent in the many-tasks limit, again establishing a connection to EBFs; we follow the approach described by Arora et al. [2019] in fixing the linear weights associated with the value function; this dramatically simplifies our analysis, while still describing practical settings in which the features and weights are trained separately as in Chung et al. [2018]. Analogous results follow when the *learning rate* β is scaled appropriately, resembling the results of Jacot et al. [2018] and Yang and Hu [2021].

Theorem 5.4.1. *For $M \in \mathbb{N}$, let $(\Phi_t^M)_{t \geq 0}$ be the solution to Equation (5.10), with each $\mathbf{w}_t^m$ for $m = 1, \ldots, M$ initialized independently from $N(0, \sigma_M^2)$, and fixed throughout training ($\beta = 0$). We consider two settings: first, where the learning rate α is scaled as $\frac{1}{M}$ and $\sigma_M^2 = 1$ for all M, and second where $\sigma_M^2 = \frac{1}{M}$ and the learning rate α is equal to 1. These two settings yield the following dynamics, respectively:*

$$\lim_{M \to \infty} \partial_t \Phi_t^M \overset{P}{=} -(I - \gamma P^\pi)\Phi_t^M \quad , \text{ and} \tag{5.12}$$

$$\lim_{M \to \infty} \partial_t \Phi_t^M \overset{D}{=} -(I - \gamma P^\pi)\Phi_t^M + R^\pi \epsilon^\top \ , \ \epsilon \sim \mathcal{N}(0, I) . \tag{5.13}$$

The corresponding limiting trajectories for a fixed initialization $\Phi_0 \in \mathbb{R}^{\mathcal{X} \times K}$ are therefore given respectively by

$$\lim_{M \to \infty} \Phi_t^M \overset{P}{=} \exp(-t(I - \gamma P^\pi))\Phi_0 \quad , \text{ and} \tag{5.14}$$

$$\lim_{M \to \infty} \Phi_t^M \overset{D}{=} \exp(-t(I - \gamma P^\pi))(\Phi_0 - (I - \gamma P^\pi)^{-1} R^\pi \varepsilon^\top) + (I - \gamma P^\pi)^{-1} R^\pi \varepsilon^\top \ , \ \epsilon \sim \mathcal{N}(0, I) . \tag{5.15}$$

Proof sketch. The key step in the proof of this result is to show that the limiting dynamics of ϕ_t^M depend on the matrix of outer products $\sum_{m=1}^M \mathbf{w}_m \mathbf{w}_m^\top$, which under the conditions provided in the theorem statement can be shown to converge to the

identity matrix. In the case of scaled initializations, we obtain the following, though the derivation in the case of scaled learning rates is similar.

$$\partial_t \Phi_t^M = (I - \gamma P^\pi)\Phi_t^M \sum_{m=1}^{M} \mathbf{w}^m (\mathbf{w}^m)^\top + \sum_{m=1}^{M} R^\pi (\mathbf{w}^m)^\top \tag{5.16}$$

$$\lim_{M\to\infty} \partial_t \Phi_t^M = (I - \gamma P^\pi)\Phi_t^M \lim_{M\to\infty} \sum_{m=1}^{M} \mathbf{w}^m (\mathbf{w}^m)^\top + \lim_{M\to\infty} R^\pi (\sum_{m=1}^{M} \mathbf{w}^m)^\top \tag{5.17}$$

$$\stackrel{D}{=} (I - \gamma P^\pi)\Phi_t^M I + R^\pi \epsilon^\top,\ \epsilon \sim \mathcal{N}(0, I). \tag{5.18}$$

In the case of zero rewards, we then get analogous dynamics on Φ_t^M as we did on the value functions in Proposition 5.3.3. □

In contrast to the case described in Section 5.3.2, this result indicates that the introduction of auxiliary tasks leads to useful environment information being incorporated into the representation. Indeed, the dynamics described above imply the following convergence result, analogous to Proposition 5.3.3.

Corollary 5.4.2. *Under the feature flow* (5.10) *with* $\mathbf{w}_t^m$ *fixed at initialization for each* $i = 1, \ldots, M$*, under either the scaled initialization or scaled learning rate assumption of Theorem 5.4.1, and under Assumption 1, for almost all initializations* Φ_0*, we have when* $R^\pi = 0$

$$d(\langle \Phi_t \rangle, \langle U_{1:K} \rangle) \to 0\,, \quad \text{as } t \to \infty.$$

The proof of this result follows straightforwardly from Theorem 5.4.1 and the proof of the convergence of value functions to the subspace spanned by $U_1, \ldots, U_k$, and can be found in Appendix C.2.

Key insight.
Under the conditions of Theorem 5.4.1 and Corollary 5.4.2, the ensemble auxiliary tasks cause the agent's representation Φ to align with EBFs.

We show in Appendix C.8 that this behaviour is observed in practice when $M \gg K$ and the value of $\mathbf{w}_t^m$ is fixed at initialization for all m. We additionally compare the representations learned when $\mathbf{w}_t^m$ is allowed to vary over training. Here we find empirically that allowing the weights to vary during training induces

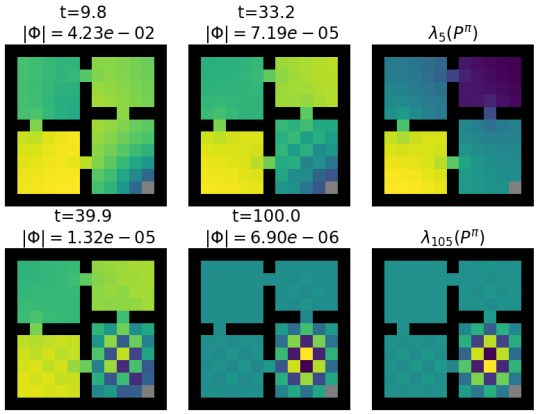


Figure 5.3: Visualization of a single column of Φ_t (i.e. feature vector) after application of the ODE in Equation (5.8) for $t \in [0, 100]$ in the four rooms environment, with $K = 10$ and $M = 20$. Early in its trajectory, ϕ_t exhibits similarity to smooth eigenfunctions (e.g. the eigenfunction corresponding to the 5^{th} greatest eigenvalue λ_5 which we plot in the top right) of P^π, but later converges to non-smooth eigenfunctions (e.g. the eigenfunction corresponding to eigenvalue λ_{105}, the most negative eigenvalue, plotted in the bottom right).

dynamics that differ from those predicted by Theorem 5.4.1 for the fixed-weights setting. To illustrate this, we follow the evolution of a single column of Φ_t, i.e. a single feature vector $\phi_t = \Phi_t[:, 1]$, trained with the ensemble prediction dynamics of Equations (5.10) & (5.11) on a simple four-rooms gridworld environment in Figure 5.3.

We visualize ϕ_t along with two illustrative eigenvectors of the transition matrix P^π, corresponding to one positive and one negative eigenvalue. We observe that while the feature ϕ_t quickly evolves to resemble the smooth eigenvector corresponding to the positive eigenvalue for small values of t, it later converges to the non-smooth eigenvector corresponding to the most negative eigenvalue of the transition matrix P^π. While we leave further analysis to future work, this example hints at an intriguing relationship between the EBFs and the joint representation dynamics.

5.4.2 Random cumulants

In the case of zero rewards, our previous results show that while from the perspective of subspaces the representation approaches the EBF subspace in Grassmann distance, in Euclidean distance the representation is approaching the zero matrix pointwise. This has important implications for the scenario of large-scale sparse-reward

environments, in which the agent may not encounter rewards for long periods of time, and indicates that the agent's representation is at risk of collapsing in such cases.

Motivated by this analysis, we consider a means of alleviating this representation collapse, by learning value functions for *randomly generated cumulants* [Osband et al., 2018, Dabney et al., 2021]. Mathematically, the agent again makes many predictions from a common representation, with each prediction indexed by $m = 1, \ldots, M$ attempting to learn the value function associated with a randomly drawn reward function $r^m \in \mathbb{R}^{\mathcal{X}}$ under the policy π. Thus, the agent's parameters are the representation Φ and a set of weights $\mathbf{w}^m$ for each prediction. The learning dynamics are then given by:

$$\partial_t \Phi_t^M = \alpha \sum_{m=1}^{M} (r^m + \gamma P^\pi \Phi_t^M \mathbf{w}_t^m - \Phi_t^M \mathbf{w}_t^m)(\mathbf{w}_t^m)^\top, \tag{5.19}$$

$$\partial_t \mathbf{w}_t^m = \beta (\Phi_t^M)^\top (r^m + \gamma P^\pi \Phi_t^M \mathbf{w}_t^m - \Phi_t \mathbf{w}_t^m)\,. \tag{5.20}$$

The main result of this section is to show that, even in the absence of reward, the limiting distribution induced by random cumulant auxiliary tasks dynamics described in Equation (5.19) is a non-zero subspace which, while dependent on the random reward function, is biased towards principal components of the transition matrix.

Theorem 5.4.3. *For fixed $M \in \mathbb{N}$, let the random rewards $(r^m)_{m=1}^M$ and weights $(\mathbf{w}^m)_{m=1}^M$ be as defined above, let $\alpha = 1$, and consider the representation dynamics in Equation (5.19), with weights fixed throughout training ($\beta = 0$). Let Σ denote the covariance matrix of the random cumulant distribution. Then*

$$\lim_{M\to\infty} \sum_{m=1}^{M} r^m (\mathbf{w}^m)^\top \overset{D}{=} Z_\Sigma \sim \mathcal{N}(0, \Sigma), \text{ and}$$

$$\lim_{M\to\infty} \Phi_t^M \overset{D}{=} \exp(-t(I - \gamma P^\pi))(\Phi_0 - (I - \gamma P^\pi)^{-1} Z_\Sigma) + (I - \gamma P^\pi)^{-1} Z_\Sigma\,.$$

As the columns of Z_Σ are mean-zero, uncorrelated, with covariance matrix Σ, the limiting distribution of each column of $\Phi_\infty = \lim_{t\to\infty} \lim_{M\to\infty} \Phi_t^M$ has covariance $\Psi\Sigma\Psi^\top$, where Ψ is the resolvent $(I - \gamma P^\pi)^{-1}$.

The proof of this result is obtained via a similar argument as in the case of Theorem 5.4.1, though care must be taken to account for the non-zero and non-uniform rewards, and can be found in Appendix C.2. Importantly, while Z_Σ may have zero *expected value* over its initialization distribution, with probability one it will not be zero. As a result, we obtain convergence to a limiting representation Φ_∞ whose value depends on the randomly initialized cumulants and the transition structure of the environment. This limiting value will thus be biased towards the lower-index eigen-basis functions, but will also have non-zero dot product with higher-index EBFs, in contrast to our results in the zero-reward setting. However, while this limiting value might result in non-zero mass assigned to lower EBFs, the *error term* of $\Phi_\infty - \Phi_t^M$ will inherit similar convergence properties as in the ensemble prediction setting, becoming dominated by the EBFs over time.

Corollary 5.4.4. *Under the feature flow* (5.19) *with* $\mathbf{w}_t^m$ *fixed at initialization for each* $i = 1, \ldots, M$ *and Assumption 1, for almost all initializations* Φ_0*, we have when* $R^\pi = 0$

$$d(\lim_{M\to\infty} \langle \Phi_t^M - \Phi_\infty \rangle, \langle U_{1:K} \rangle) \to 0\,, \quad \text{as } t \to \infty.$$

Theorem 5.4.3 indicates that the left-singular vectors of $\Sigma^{1/2}\Psi$ (or equivalently, the right-eigenvectors of $\Psi\Sigma\Psi^\top$) are key to understanding the effects of random cumulants on representations; we introduce the term *resolvent singular basis functions* (RSBFs) to refer to these vectors in the canonical case $\Sigma = I$.

Key insight.
With random cumulant auxiliary tasks, under the assumptions of Theorem 5.4.3 and Corollary 5.4.4, the distribution of the limiting representation does not collapse, and is characterized by the RSBFs of P^π, while the trajectory it follows to reach this subspace is determined by the EBFs of P^π.

These decompositions of P^π bear deep connections to prior work on feature learning. EBFs correspond to the eigendecomposition of the successor representation, which can be explicitly related to the proto-value functions described by Mahadevan [2009] when the transition matrix P^π corresponds to that of a random walk policy

[Machado et al., 2018b]. For symmetric P^π we obtain an additional correspondence between EBFs and RSBFs, though we note that when P^π is not symmetric the RSBFs may differ from both the EBFs and the singular value decomposition of the transition matrix P^π. We provide further discussion of RSBFs and comparisons against existing concepts in feature selection in Appendix C.7.

The subspace given by the RSBFs has a further appealing connection to General Value Functions (GVFs). We provide additional details in Appendix C.6, where we show that RSBFs can be viewed as Bayes-optimal features in the sense that they minimize the expected value function approximation error given an isotropic Gaussian prior on an unknown reward function.

5.4.3 Analysis of additional auxiliary tasks

The infinite-task limit simplifies the analysis of a broad range of auxiliary tasks, and analogous results to Theorem 5.4.1 can be easily derived for other tasks that involve value prediction for some discount factor and policy. We provide a summary of these results in Table 5.1, including their full statements and derivations in Appendix C.3. We consider two additional classes of auxiliary task: predicting the values of multiple policies [Dabney et al., 2021], and predicting the value function of the current policy under multiple discount factors [Fedus et al., 2019a].

Under the **multiple policies** auxiliary task, the agent's objective is to learn a set of value functions $V^1, \ldots, V^M$ such that $V^i(x) = \mathbb{E}_{\pi_i}[R^{\pi_i}(x) + \gamma P^{\pi_i} V^i(x)]$. We consider an ensemble prediction variant of this objective, where given a fixed set of k policies, we train an ensemble of M predictors $V^{1,1}, \ldots, V^{m,1}, \ldots, V^{1,k}, \ldots, V^{m,k}$, where $m = \frac{M}{k}$ and the value function $V^{i,j}$ is trained on policy π_j.

Under the **multiple discount factors** auxiliary task, the agent's objective is analogously to find $V^i(x) = \mathbb{E}_{\pi_i}[R^\pi(x) + \gamma_i P^\pi V^i(x)]$ for $\gamma_i \in \gamma_1, \ldots, \gamma_k$. As with the multiple policies auxiliary task, we assign to each discount factor objective multiple prediction heads $V^{1,1}, \ldots, V^{m,1}, \ldots, V^{1,k}, \ldots, V^{m,k}$, where $m = \frac{M}{k}$ and the value function $V^{i,j}$ is trained on discount factor γ_j.

In both cases, under the conditions of the previous theorems, the dynamics of the ensemble converge to the dynamics induced by the mean of the set of auxiliary tasks, implying the counter-intuitive result that training with multiple auxiliary tasks does not provide additional utility over the single task setting. This apparent shortcoming stems from the large number of auxiliary prediction objectives all operating on the same 'feature subspace' of $\mathbb{R}^d$, essentially forcing any dimension of that subspace to fit the mean target value. It can be modified to more closely resemble practical settings by ensuring that the weights corresponding to each auxiliary task π_i or γ_i are initialized in *orthogonal subspaces*, so that the vector space V in which the representation evolves can be decomposed as the direct sum of the subspaces V_i associated with each task i, $V = \oplus_{i\in[1,k]} V_i$. In this case, we obtain an analogous decomposition of the representation Φ and its corresponding dynamics, obtaining convergence to a direct sum of the limiting representation of each task. This suggests that the benefits of auxiliary tasks might be maximized by appropriate initialization schemes which encourage the representations learned for each task to be (linearly) independent.

5.4.4 Dynamics in the infinite-width limit

So far, the analysis of this chapter has studied an idealized feature-learning model where the features are represented by an extremely large matrix, and for which an update to the feature vector of one state will not influence the features of other states. This simplifies our analysis, but diverges from the function approximation regimes seen in practice, where a neural network is often used to construct the feature representation. We take a step towards the deep RL setting by considering a limiting case of DNN function approximation.

Recent results on neural networks in the limit of infinite width reveal that their learning dynamics are determined by a kernel, denoted the *neural tangent kernel* (NTK) [Jacot et al., 2018], described in more detail in Section 2.2.2. This allows us to apply the analysis of the previous sections to the infinite-width limit of neural

networks, for which we obtain the following dynamics.

$$\partial_t V_t = \Phi_{\Theta_\infty^{(L)}}(\gamma P^\pi - I)V_t \,. \tag{5.21}$$

We now see that the kernel matrix $\Phi_{\Theta^{(L)}}$ influences the trajectory of the value function, rather than solely P^π as previously. We can now directly quantify how the auxiliary losses influence the trajectory of the value function in the infinite-width regime, as we describe in the following theorem.

Theorem 5.4.5. *Let $f = (f^1, \ldots, f^M) : \mathcal{X} \to \mathbb{R}^M$ be computed by an L-layer neural network with layer widths $n_1, \ldots, n_L = M$, parameters θ, and Lipschitz nonlinearity σ. Let f_t^1 follow the Bellman-error minimizing flow of Eq (5.2), and let the joint loss δ be of the form*

$$\delta(f) = \delta_1(f^1) + \delta_2(f^2, \ldots, f^k) \,.$$

Under the conditions of Theorem 2 of Jacot et al. [2018], with the limiting NTK matrix $\Phi_{\Theta_\infty^{(L)}}$ defined therein, the dynamics $\partial_t f_t^1$ are independent of the values of $f_t^2, \ldots, f_t^k$. Thus, for any set of auxiliary tasks δ_2, number of heads $k > 1$, and auxiliary head values $f_t^2, \ldots, f_t^m$,

$$\lim_{n_1, \ldots n_{L-1} \to \infty} \partial_t f_t^1 = \Phi_{\Theta_\infty^{(L)}} \nabla_{f^1} \delta_1(f_t^1) \,. \tag{5.22}$$

This result highlights the importance of the shared feature dynamics between the network outputs in order for auxiliary tasks to influence value function learning. The initialization and dynamics of the NTK regime result in each head of the network evolving independently; essentially, each of the auxiliary tasks is operating on a mutually-orthogonal subset of the feature space, and as a result incorporating auxiliary tasks will not accelerate learning of the value function. This is in contrast to most DNN architectures used in deep RL, which tend to be relatively small and so exhibit significant interference (both positive and negative) between tasks. Two notable exceptions to this observation are categorical distributional reinforcement learning, where the softmax operator induces dependence between the different heads' loss functions, and random ensemble mixture (REM) [Agarwal et al., 2020],

where a random mixture is taken over an ensemble of value prediction heads before applying the TD loss. The analysis of the effect of this family of auxiliary tasks on representation dynamics in wide neural networks requires additional mathematical tools to account for their nonlinear output layer, however, and falls outside the scope of this chapter. Other parameterizations such as those studied by Yang and Hu [2021] may further yield more interesting feature-learning dynamics even for the classes of auxiliary tasks studied in this chapter.

5.5 Experiments

In this section, we complement the previous theoretical results with empirical investigations in both tabular and deep reinforcement learning settings. These empirical results will motivate the directions taken in Chapter 6, where we will further explore both the sparse- and dense-reward setting.

5.5.1 Feature generalization across the value-improvement path

Having established connections between the representations induced by auxiliary tasks and several decompositions of the environment transition operator, we now turn to the question of how useful these representations are to a reinforcement learning agent. In particular, we address how well representations learned under one policy *generalize* under the policy improvement step to approximate future value functions, with particular attention paid to EBFs and RSBFs, the decompositions that feature in our earlier analysis.

To address this question empirically we run tabular policy iteration on a stochastic chain MDP, yielding a sequence of policies $(\pi_j)_{j=1}^J$ and associated value functions $(V_j)_{j=1}^J$. We then compute EBFs and RSBFs associated with P^π, and compute the acute angle between V_j and the subspace spanned by these features, for each $j \in [J]$; this is in fact equal to a generalization of the Grassmann distance for subspaces of unequal dimension [Ye and Lim, 2016]. We also compare against

a baseline of isotropic randomly-generated features. Full experimental details are provided in Appendix C.9.

Results are given in the top row of Figure 5.4 for the case of four features; each individual heat map plots Grassmann distances, with rows indexing the policy that generated the features, and columns indexing the policy yielding the target value function. In general, the RSBFs provide better transfer across policies in the improvement path relative to random features and EBFs. For times $j, j' \in [J]$, we observe that the Grassmann distance between the RSBFs of P^{π_j} and the value function of j', $V^{\pi_{j'}}$, increases as $|j - j'|$ does.

We also evaluate transfer when the vector V^{π_j} is added to the set of features, in the bottom row of Figure 5.4. This contains the subspace to which the value functions described in Proposition 5.3.3 converge, as the limiting solutions can be described as being of the form $V^{\pi_j} + u$ for $u \in \langle U_{1:K} \rangle$. This results in a significant improvement to the ability of the EBFs to fit future value functions, suggesting that representation learning objectives which *only* seek to incorporate information about the transition dynamics, without including any reward information, may be sub-optimal for representing value functions of interest. Surprisingly, we find that this addition results in the EBFs for π_j outperforming RSBFs specifically in predicting $V^{\pi_{j+1}}$. This can be observed in the upper off-diagonal the EBF plot

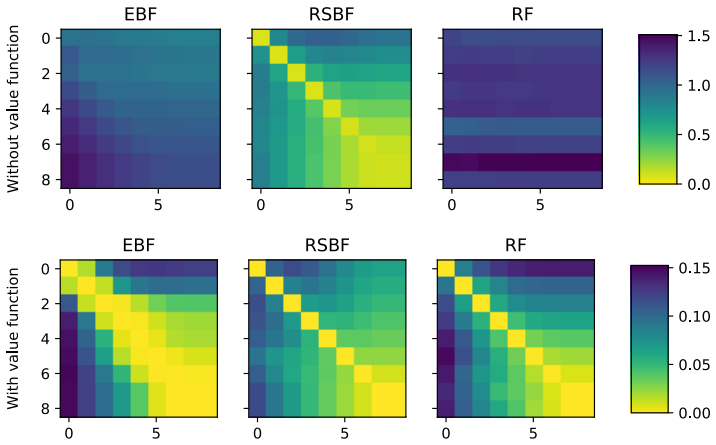


Figure 5.4: Transfer of EBFs, RSBFs, and RFs across the value-improvement path of a chain MDP, with and without the value function as an additional feature.

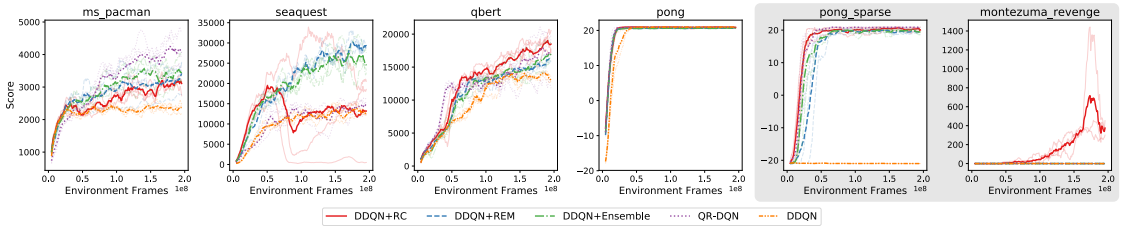


Figure 5.5: Learning curves for DDQN, DDQN+RC, DDQN+Ensemble, DDQN+REM, and QR-DQN agents on several dense reward ALE environments. Two games with sparse rewards are shown in the shaded box.

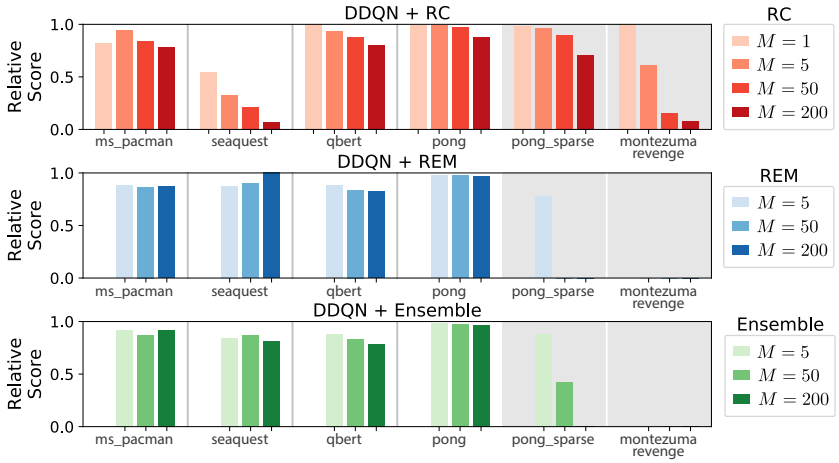


Figure 5.6: Sweep over number of auxiliary heads for RC, REM and Ensemble. Each bar corresponds to a single game, presented in the same order as Figure 5.5. Relative score is the per-game score divided by the maximum over all algorithms and hyperparameters.

in Figure 5.4. We conclude that the dynamics induced by TD updates may be particularly beneficial to transfer between policies in the value-improvement path, and further study of this phenomenon is a promising avenue for future work.

5.5.2 Auxiliary tasks for large-scale environments with sparse rewards

We now turn our attention to deep reinforcement learning, particularly in the context of environments with sparse reward structure. Motivated by the theoretical results obtained in earlier sections, we study the effects of a variety of auxiliary tasks in this setting; our analysis indicates that random cumulants may be particularly effective in preventing representation collapse in such environments. Concretely, this section will seek to evaluate whether auxiliary tasks that incorporate non-zero

reward signals can improve performance in sparse-reward tasks; Chapter 6 will then investigate the mechanisms behind this in more depth.

Hypothesis 5.5.1: incorporating non-zero reward in auxiliary tasks will improve the learned representation, and thereby improve performance, by preventing representation collapse in deep RL agents.

To evaluate Hypothesis 5.5.1, we empirically evaluate the performance of a variety of agents on a representative subset of Atari environments. We modify a Double DQN agent [Van Hasselt et al., 2016] with a variety of auxiliary tasks, including random cumulants (RC) [Dabney et al., 2021], random ensemble mixtures (REM) [Agarwal et al., 2020], an ensembling approach [Anschel et al., 2017], and also compare with QR-DQN, a distributional agent [Dabney et al., 2018]. Full details of these agents, including specific implementation details for deep RL versions of these auxiliary tasks, are given in Appendix C.9.

We evaluate these agents on a series of Atari games from the Arcade Learning Environment [Bellemare et al., 2013, Machado et al., 2018a], comprising Montezuma's Revenge, Pong, MsPacman, Seaquest, and Q*bert. In addition, we evaluate on a more challenging, sparse reward, version of Pong in which the agent does not observe negative rewards.[2]

Figure 5.5 shows the main results from these experiments. Recall from Section 5.4 that the random cumulant auxiliary task causes the agent's representation to converge to the RSBFs of P^π in idealized settings. We hypothesize that this auxiliary task will therefore improve agent performance over ensemble-based auxiliary tasks in sparse-reward environments. Our empirical results support our conjecture, with the random cumulant agent (DDQN+RC) generally performing well in the sparse-reward environments. Of particular note is the strong performance in Montezuma's Revenge. We expected reduced performance for DDQN+RC in the dense-reward games, but were surprised to observe improved performance here as well, with the

[2]We attempted a similar modification of the other three dense reward games, but found no agent or configuration that was able to successfully learn on them. Full details, along with hyperparameters and results on these unsuccessful modifications, are given in Appendix C.9.

exception of Seaquest. Finally, Figure 5.6 shows the result of a hyperparameter sweep over the number of auxiliary task heads, revealing relevant differences in the three methods considered. Overall, we find that random cumulants are a promising auxiliary task specifically in sparse-reward environments. In the following chapter, we will investigate an approach which adapts the random cumulant task to avoid interference in dense-reward tasks like Seaquest while preserving its ability to prevent representation collapse in sparse-reward environments.

5.6 Conclusions

This chapter has introduced a framework based on learning dynamics to analyze representations in reinforcement learning. This led to a variety of theoretical results concerning learning with and without the presence of auxiliary tasks, as well as several straightforward models for studying representation learning empirically. In particular, we neatly characterized the *trajectory* taken by value functions and, in some cases, representations under TD learning dynamics. With this, we were able to thoroughly test a new hypothesis on the effectiveness of particular auxiliary tasks in sparse-reward environments, which led to improved understanding of representation learning in RL, as well as practical modifications to deep RL algorithms.

There are many natural follow-up directions to this work, some of which we will explore in the following chapters. One direction is to further develop the theory associated with the learning dynamics perspective, in order to (i) understand how additional types of auxiliary tasks, in particular auxiliary tasks that do not correspond to value functions, affect the representations in the learning models developed in this chapter, (ii) extend the learning models themselves to incorporate further aspects of large-scale learning scenarios, such as sample-based learning and state-visitation distribution corrections, and (iii) investigate other common learning dynamics, such as gradient TD methods [Sutton et al., 2008] or policy gradient updates. In some cases, as with non-uniform state distributions, such extensions are straightforward. However, other less trivial extensions are crucial to improve the applicability of this model to practical settings. The current

formulation of the learning dynamics model does not capture several important factors of practical deep RL training paradigms, leaving a gap between the exact, continuous-time representation updates of Theorem 5.4.1 and those of finite-width deep neural networks whose architectures endow the representation dynamics with particular inductive biases.

The following chapters will bridge this gap to extract useful methodological implications of this framework. First, Chapter 6 will capitalize on our observation that sparse-reward environments encourage feature collapse to study the mechanisms by which networks progress along the value improvement path. It will propose an auxiliary task which avoids the pitfalls of the RCDQN objective, which can hinder progress in some dense-reward environments, and to develop a more fine-grained view on the impact of non-stationarity on the plasticity of the learned representation. Next, Chapter 7 will study the implications of Theorem 5.4.1 and Proposition 5.3.3 as they relate to generalization and interference. This discussion will take a complementary view of the presentation in this chapter by considering not the subspace spanned by the predicted value functions, but rather the properties of the approximation error incurred by the agent.

6
Capacity loss

Contents

6.1 Introduction 115
6.2 Learning capacity in neural networks 119
6.3 InFeR: mitigating capacity loss with feature regularization 128
6.4 Conclusions 133

6.1 Introduction

The reinforcement learning problem presents a number of difficulties not present in supervised learning. Chief among these is the *non-stationarity* of the learning objective. RL agents must solve a sequence of similar prediction problems as they iteratively improve their prediction accuracy and their policy [Dabney et al., 2021], and sufficient improvement in each subproblem in this sequence is necessary to progress to the next subproblem. Ideally a solution to one subproblem should *generalize* to the subsequent one, in the sense that the learned representation should enable rapid adaptation. At minimum, fitting one prediction problem should not hinder performance on the next. This challenge was alluded to in Chapter 5; we now confront it explicitly, focusing particularly on its effect in RL agents using deep

neural networks as function approximators. Prior works have shown that the early training period of a network is critical for its ability to achieve optimal performance on a task [Frankle et al., 2020b, Achille et al., 2018], and that early exposure to diverse data is crucial for optimal generalization performance [Ash and Adams, 2020, Berariu et al., 2021]. This suggests that the non-stationary target functions prevalent in reinforcement learning may be particularly ill-suited to function approximation by DNNs trained with gradient-based optimization, particularly in sparse-reward environments where the network receives no learning signal in this critical phase. Indeed, several prior works studying the effect of re-initializing network parameters in reinforcement learning have found that this enables agents to break through plateaus [Fedus et al., 2019b] and improve generalization performance [Igl et al., 2021]. This chapter will study one mechanism driving these phenomena, and lay the groundwork for a more thorough empirical analysis of the effect of prediction targets on generalization in Chapter 7.

6.1.1 Motivating example: the zero function

We follow Chapter 5 in framing representation-learning in terms of a learned feature layer $\phi_{\boldsymbol{\theta}}(\mathbf{x})$ which is composed with some simple (usually linear) output function to predict state-action values. Previously, we restricted ourselves to settings where the representation's dynamics did not depend on the particular value of the linear map $\mathbf{w}$, as the ensuing dynamics became intractable to analyze theoretically when Φ and $\mathbf{w}$ were coupled. We now take an empirical approach to study what happens when the dynamics of Φ take more complex forms. To do so, we will focus on the location of a given set of parameters in the optimization landscape. This can provide some information not available in the outputs of a feature layer, as there are many different sets of parameters which correspond to the same function outputs, but which may occupy locations in the loss landscape with very different properties, making some easier to use as a starting point for learning than others.

To provide a concrete illustration of this, consider a single hidden layer ReLU network. We let $\mathbf{x}$, $\mathbf{x} \in \mathbb{R}^d$ denote the observation, and $\boldsymbol{\theta} \in \mathbb{R}^{k \times d}$ network's

first layer weights. We then write $\phi_{\boldsymbol{\theta}}(\mathbf{x}) = \sigma(\boldsymbol{\theta}^\top \mathbf{x})$ where $\sigma(y) = \max(0, y)$ is applied entry-wise to the vector $\boldsymbol{\theta}^\top \mathbf{x}$. The output of the network is then $f_{\boldsymbol{\theta}}(\mathbf{x}) = \langle \mathbf{w}, \phi_{\boldsymbol{\theta}}(\mathbf{x}) \rangle$. Suppose we have encountered the (unlikely) scenario that all of the parameters $\boldsymbol{\theta}$ are initialized to have negative value, the input distribution is only supported on non-negative vectors, and $\mathbf{w}$ is initialized to be exactly zero. Then, recalling the framework of Chapter 5, any regression objective with target $\mathbf{y}$ will face the following dynamics.

$$\nabla_{\boldsymbol{\theta}} (\mathbf{y} - f_{\boldsymbol{\theta}}(\mathbf{x}))^2 = (\mathbf{y} - f_{\boldsymbol{\theta}}(\mathbf{x})) \cdot (\mathbf{w} \cdot \mathbf{0}) = \mathbf{0} \tag{6.1}$$

$$\nabla_{\mathbf{w}} (\mathbf{y} - f_{\boldsymbol{\theta}}(\mathbf{x}))^2 = (\mathbf{y} - f_{\boldsymbol{\theta}}(\mathbf{x})) \cdot \phi_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{0} \tag{6.2}$$

As a result, the gradient descent trajectory is locked in at the current parameters and will not be able to escape them. In contrast, if $\boldsymbol{\theta}^\top \mathbf{x}$ contains non-negative elements, then even if $\mathbf{w}$ is initialized to equal zero and the network still outputs the same function value, the gradients for $\mathbf{w}$ will be non-zero and so the gradient descent trajectory will be able to escape this initialization. Thus, under one parameterization of the zero function the network is never able to update its predictions, while under the other it can do so with relative ease. Relating this back to the analysis of the previous chapter, this condition is equivalent to saying that we would like at least one of $\partial_t \boldsymbol{\theta}_t$ and $\partial_t \mathbf{w}_t$ to be non-zero. This failure mode can be considered a special case of *mutually frozen weights* [Zilly et al., 2021], a more general failure mode whereby saturated units prevent networks from making learning progress.

6.1.2 Contributions

The principal hypothesis of this chapter is that over the course of training, deep RL agents lose some of their capacity to quickly fit new prediction tasks, and in extreme cases this capacity loss prevents the agent entirely from making learning progress. In other words, capacity loss results in representations which *fail to generalize* to the value functions induced by new policies and new TD targets in the value improvement path. We will show that the ability of deep RL agents to fit new

target functions declines over the course of training in several environments from the Atari suite [Bellemare et al., 2013] and on a non-stationary supervised prediction tasks. We take a deeper look at the representation collapse phenomenon, where the feature outputs for every state in the environment inhabit a low-dimensional – or possibly even zero – subspace, extending the empirical analysis of Chapter 5. Finally, we provide evidence that representation collapse is a key factor in agents' failure to make performance improvements in sparse-reward environments. We saw in the previous chapter that predicting *random cumulants* can prevent feature collapse in sparse-reward environments; however, it can also lead to interference in dense-reward environments, harming performance.

To address this limitation we propose a simple regularization technique, Initial Feature Regularization (InFeR), to prevent capacity loss in both dense- and sparse-reward environments. In this approach, we regress a set of feature projection heads to their values at initialization. While the regression targets used in InFeR can be viewed as an auxiliary task, the method does not incorporate any additional environment information into the learning objective. This allows us to isolate the effect of capacity loss on agent performance without introducing confounding from the interaction between the auxiliary environment information and the agent's representation. We find that this regularization scheme mitigates capacity loss in sequential supervised prediction tasks, and that RL agents trained with InFeR avoid egregious cases of representation collapse in sparse reward environments. We further show that InFeR works by regularizing the network's learning dynamics, and conduct ablation studies to better understand this mechanism.

One striking take-away from these results is that agents trained on so-called 'hard exploration' games such as Montezuma's Revenge can attain significant improvements over existing competitive baselines *without* using smart exploration algorithms. This suggests that the poor performance of deep RL agents in sparse-reward environments is not *solely* due to inadequate exploration, but rather also in part due to poor representation learning as the network 'overfits' to predicting the zero function. There are thus two levers one can pull to improve

performance: increasing the amount of reward signal the agent encounters via improved exploration, and ensuring that it is able to effectively update its predictions when it does receive reward signal. These two levers produce complex feedback loops in practical environments: more accurate predictions produce more informative behaviours, and more informative behaviours provide information needed to improve the network's predictions. The sparse reward settings studied in this chapter highlight the necessity of network plasticity in enabling this virtuous cycle. Chapter 7 will study a more nuanced form of overfitting to early targets that arises even in dense-reward environments, extending the insights from this chapter to a broader range of settings.

6.2 Learning capacity in neural networks

Each time an RL agent discovers a new source of reward in its environment or improves its ability to obtain this reward, the value function that it seeks to predict changes. Over the course of learning, a value-based RL agent attempts to solve a long sequence of target prediction problems, though in the case of value iteration-style algorithms such agents many only partially solve each problem before the next is constructed. Studies of neural networks in supervised learning suggest that this sequential fitting of new targets may be harmful to a neural network's ability to adapt to new objectives [Ash and Adams, 2020]. This presents a concern for deep RL, where networks are trained to fit a constantly-changing target and may need to quickly make significant changes to their predictions even late in the training process. In this section, we show that training on a sequence of prediction targets can indeed lead to a reduced ability to fit new objectives in deep neural networks, a phenomenon that we term *capacity loss*, and confirm that capacity loss can arise in value-based deep RL agents. Further, we provide evidence that an agent's inability to quickly update its value function to distinguish states presents a barrier to performance improvement in deep RL agents trained on environments from the Atari suite.

6.2.1 Target-fitting capacity

The parameters of a neural network determine not just the network's current outputs, but also how these outputs will evolve over time via the magnitude and structure of its gradients and the curvature of the loss landscape. This evolution determines the *capacity* of the network to learn to predict new targets by following an optimization trajectory. We therefore view the agent's representation in terms of the optimization dynamics that it induces. In particular, we are interested in identifying when an agent's current parameters are flexible enough to allow it to perform gradient updates that meaningfully change its predictions based on new reward information in the environment or evolving bootstrap targets, a notion formalized in the following definition.

Definition 6.2.1 (Target-fitting capacity)**.** *Let $P_X \in \mathscr{P}(X)$ be some distribution over inputs X and $P_{\mathcal{F}}$ a distribution over a family of functions $\mathcal{F}$ with domain X. Let $\mathcal{N} = (g_\theta, \theta_0)$ represent the pairing of a neural network architecture with some initial parameters θ_0, and $\mathcal{O}$ correspond to an optimization algorithm. We measure the capacity of $\mathcal{N}$ under the optimizer $\mathcal{O}$ to fit the data-generating distribution $\mathcal{D} = (P_X, P_{\mathcal{F}})$ as follows:*

$$\mathcal{C}(\mathcal{N}, \mathcal{O}, \mathcal{D}) = \mathbb{E}_{f \sim P_{\mathcal{F}}}[\mathbb{E}_{x \sim P_X}[(g_{\theta'}(x) - f(x))^2]] \quad \textit{where } \theta' = \mathcal{O}(\theta_0, P_X, f)\,. \tag{6.3}$$

This definition of capacity measures the ability of a network to reach a new set of targets within a limited optimization budget from its current parameters and optimizer state. The choice of optimization budget and target distribution are left as free variables, and different choices result in different notions of capacity. In reinforcement learning we ultimately care about the network's ability to fit its Bellman targets quickly, particularly to the extent that it allows the greedy policy to obtain high reward. However, only evaluating a network's ability to fit its Bellman targets within a small budget will not necessarily be a useful measure of capacity: for example, a network which can only output the zero function will

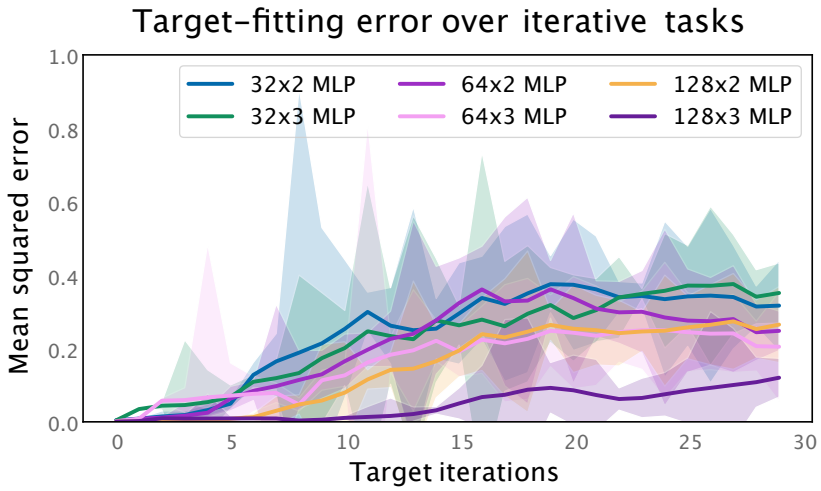


Figure 6.1: Deeper analysis on a sequential MNIST prediction setting showing that target-fitting capacity sees the greatest reduction in low capacity networks with ReLU units.

attain low Bellman error immediately on a sparse-reward environment, but will fail to produce useful updates to improve its policy. Our evaluations of this measure will assign longer training budgets and use target functions that are independent of the current network parameters to avoid these pathologies; the effect of this choice is explored further in Appendix D.2.2.

The process of training a neural network to fit a set of labels must by necessity change some properties of the network. Works studying the information bottleneck principle [Tishby and Zaslavsky, 2015] for example, identify a compression effect of training on the latent representation, where inputs with similar labels are mapped to similar feature vectors. This compression can benefit generalization on the current task, but may make learning more difficult if in the future the network must distinguish between these inputs. In the face of the rapidly-changing nature of the targets used in value iteration algorithms, this compression has the potential to harm the learning process by impeding the network's ability to fit new targets rather than to help it. This motivates the first hypothesis of this section.

Hypothesis 6.2.1: networks trained to iteratively fit a sequence of dissimilar targets will lose their capacity to fit new target functions within a fixed optimization budget compared to their capacity at initialization.

To evaluate Hypothesis 6.2.1, we construct a series of iterative prediction problems on the MNIST data set, a widely-used computer vision benchmark which consists of images of handwritten digits and corresponding labels. We first fit a series of labels computed by a randomly initialized target neural network f_θ: we transform input-label pairs (x, y) from the canonical MNIST dataset to $(x, f_\theta(x))$, where $f_\theta(x)$ is the network output. To generate a new task, we simply reinitialize the target network. Given a target function, we then train a student neural network for a fixed budget from the parameters obtained at the end of the previous iteration (using a random initialization for the first iteration), and repeat this procedure of target initialization and training thirty times. We use a subset of MNIST inputs of size 1000 to reduce computational cost. In these experiments we focus only on the ability of the network to fit its training data; a study of the effect of prediction targets on generalization is deferred to Chapter 7. We provide additional details in Appendix D.2.1.

In Figure 6.1 we see that the networks trained on this task exhibit decreasing ability to fit later target functions under a fixed optimization budget. This effect is strongest in the smaller networks, matching the intuition that solving tasks which are more challenging for the network will result in greater capacity loss. We consider two other tasks in Appendix D.2.2, as well as a wider range of architectures, obtaining similar results. We find in this broader set of evaluations that sufficiently over-parameterized networks (on the order of one million parameters for a task with one thousand data points) exhibit positive forward transfer; however, models which are not over-parameterized relative to the task difficulty consistently exhibit increasing error as the number of targets trained on grows.

This raises a question: are the deep neural networks used by value-based RL agents on popular benchmarks in the over- or under-parameterized regime? The examples here differ in an important way from the value iteration methods common in deep RL: we run our optimizer on each task long enough for the loss to converge before turning to the next task. In RL problems, the network will not have converged to the value of the current policy before the changes to the value function induce a

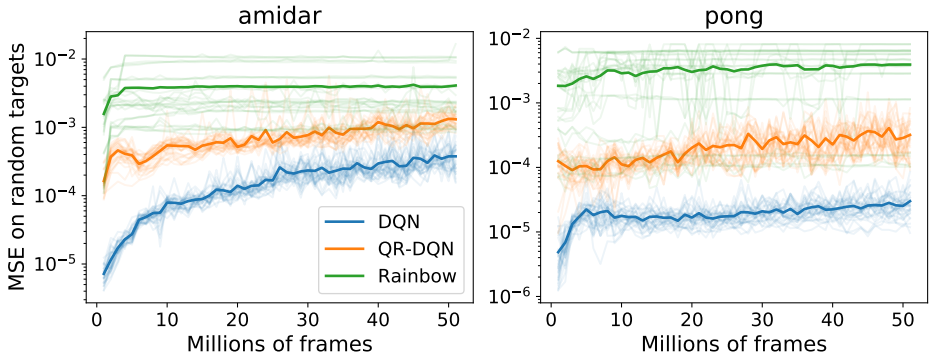


Figure 6.2: Neural networks exhibit a decline in their ability to fit random network outputs over the course of training in two demonstrative Atari environments.

policy improvement step. We therefore turn our attention to reinforcement learning, to evaluate whether the phenomena observed in the idealized supervised learning setting also hold in value-based RL agents.

Hypothesis 6.2.2: the non-stationary prediction problems in value-based deep RL also result in capacity loss.

To evaluate Hypothesis 6.2.2, we train network checkpoints stored over the course of training to fit randomly generated target functions. We provide full details of this procedure in Appendix D.3. We generate target functions by randomly initializing a neural network with an identical architecture to the agent's, and use the outputs of this network as targets for regression. We then load initial parameters from an agent checkpoint at some time t, sample inputs from the replay buffer, and regress on the random target function evaluated on these inputs. We then evaluate the mean squared error after training for fifty thousand steps. We consider a DQN [Mnih et al., 2015], a QR-DQN [Dabney et al., 2018], and a Rainbow agent [Hessel et al., 2018]. We observe in all three cases that as training progresses agents' checkpoints obtain increasing error after training to fit randomly generated targets; due to space limitations we only show two representative environments where this phenomenon occurs in Figure 6.2, and defer the full evaluation to Appendix D.3.3.

Key insight.
When neural networks are trained to predict a sequence of challenging target functions, they get progressively worse at fitting these targets. At least some RL settings exhibit this property, resulting in *capacity loss* over the course of training.

6.2.2 Representation collapse and performance

The notion of capacity in Definition 6.2.1 measures the ability of a network to *eventually* represent a given target function. This definition reflects a number of intuitions about capacity: that networks which are good at fitting only a narrow range of targets should have lower capacity than more adaptable ones, and that capacity should decrease over the course of training as the network becomes more specialized. However, this quantity is computationally expensive to compute and is dependent on the choice of target function class.

We now introduce an alternative measure of capacity which captures a network's ability to quickly adapt to changes in the target function, while being significantly cheaper to compute than the more precise measure introduced in Definition 6.2.1. We call this notion of capacity the feature rank, as it corresponds to an approximation of the rank of a feature embedding. Intuitively, the feature rank measures how easily states can be distinguished by updating only the final layer of the network. This notion of state similarity is particularly relevant to sparse-reward environments, where policy improvement depends on the agent's ability to distinguish a handful of rewarding states from the vast non-rewarding majority. It also bears close resemblance to the implicit underparameterization phenomenon [Kumar et al., 2021] studied previously, allowing us to evaluate how well these prior notions of capacity capture a network's ability to change its predictions and improve its performance.

Definition 6.2.2 (Feature rank)**.** *Let $\phi : \mathcal{X} \to \mathbb{R}^d$ be a feature mapping. Let $\mathbf{X}_n \in \mathcal{X}^n$ be a set of n states in $\mathcal{X}$ sampled from some fixed distribution P. Fix $\varepsilon \geq 0$, and let $\phi(\mathbf{X}_n) \in \mathbb{R}^{n \times d}$ denote the matrix whose rows are the feature embeddings of*

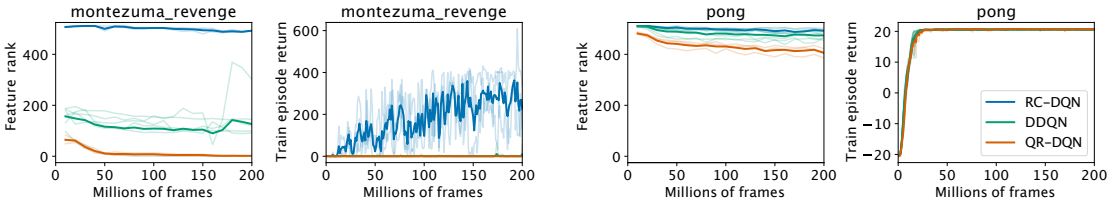


Figure 6.3: Feature rank and performance over the course of training. We observe that feature rank is higher for environments and auxiliary tasks which provide denser reward signals than for sparse reward problems.

states $x \in \mathbf{X}_n$. *Let* $\text{SVD}(M)$ *denote the multiset of singular values of a matrix* M. *Then the feature rank of* ϕ *given input distribution* P *is defined to be*

$$\rho(\phi, P, \epsilon) = \lim_{n\to\infty} \mathbb{E}_{\mathbf{X}_n \sim P}[|\{\sigma \in \text{SVD}\left(\frac{1}{\sqrt{n}}\phi(\mathbf{X}_n)\right) | \sigma > \varepsilon\}|] \tag{6.4}$$

for which a consistent estimator can be constructed as follows

$$\hat{\rho}_n(\phi, \mathbf{X}, \epsilon) = |\{\sigma \in \text{SVD}\left(\frac{1}{\sqrt{n}}\phi(\mathbf{X})\right) | \sigma > \varepsilon\}| \,. \tag{6.5}$$

The feature rank is equal to the dimension of the subspace spanned by $\phi(\mathcal{X}) = \{\phi(x) \mid x \in \mathcal{X}\}$ when $\varepsilon = 0$ and the state space $\mathcal{X}$ is finite. For $\epsilon > 0$, it throws away small components of the feature matrix. We show that ρ is well-defined and that $\hat{\rho}_n$ is a consistent estimator in Appendix D.1.1. Our analysis of the feature rank resembles that of Kumar et al. [2021], but differs in two important ways: first, our estimator does not normalize by the maximal singular value. This allows us to more cleanly capture *representation collapse*, where the network features, and thus also their singular values, converge to zero. Second, we are interested in the capacity of agents with unlimited opportunity to interact with the environment, rather than in the data-limited regime. We compare our findings on feature rank against the *srank* used in prior work in Appendix D.2.2.

We proceed to empirically evaluate the feature rank of various deep RL agents trained on games from the Atari suite to study its evolution over the course of training. We train a double DQN (DDQN) agent, a quantile regression (QRDQN) agent, and a double DQN agent with an auxiliary random cumulant prediction task (RC-DQN) [Dabney et al., 2021], on environments from the Atari suite, then evaluate $\hat{\rho}_n$ with $n = 5000$ on agent checkpoints obtained during training. We revisit two of

the environments studied in Chapter 5: Montezuma's Revenge (sparse reward), and Pong (dense reward), deferring two additional environments, a sparsified version of Pong in which the agent does not receive negative reward when its opponent scores, and Seaquest (dense reward, but more challenging than Pong), to Appendix D.3.2. We run 3 random seeds on each environment-agent combination.

We visualize agents' feature rank and performance in Figure 6.3. Our findings confirm that the RC-DQN objective prevents representation collapse, as previously conjectured. More generally, non-trivial prediction tasks, either value prediction in the presence of environment rewards or auxiliary tasks, lead to higher feature rank. Unlike in the case of target-fitting capacity, feature rank does not decline monotonically, indicating that it measures a subtly different notion of capacity than simply an agent's ability to fit new target functions. This subtlety will be explored further in Chapter 7, where it will be interpreted as incorporating a particular inductive bias into the network structure. In Montezuma's Revenge, the higher feature rank induced by RC-DQN corresponds to higher performance, but as we saw in Chapter 5, this auxiliary loss can have a detrimental effect on learning progress in complex, dense-reward games such as Seaquest, presumably due to interference between the random rewards and the true learning objective. Unlike in target-fitting capacity, we only see a consistent downward trend in sparse-reward environments, where a number of agents, most dramatically QRDQN, exhibit representation collapse. We discuss potential mechanisms behind this trend in Appendix D.1.2. We conjecture that in practice, the feature rank is a better measure of the inductive bias of the network than it is a measure of capacity, but that it can accurately and cheaply detect extreme instances of feature collapse which may impede learning.

While the many moving parts in deep RL algorithms make it difficult to isolate the causal effect of a single representation property on performance, Figure 6.4a reveals a correlation between learning progress and feature rank on challenging games where agents fail to achieve human-level performance. We see this correlation in both a Rainbow [Hessel et al., 2018] agent, and an agent trained with the regularizer we will introduce in the coming section whose precise form is not relevant

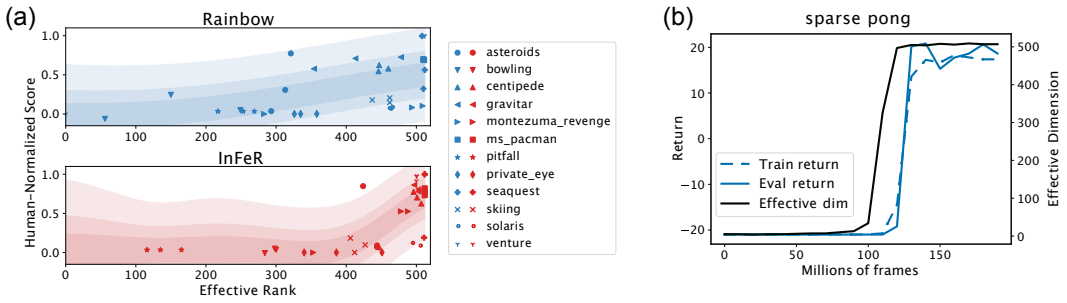


Figure 6.4: **(a)**: Agent feature rank vs human-normalized score in games where Rainbow does not achieve superhuman performance. While feature rank does not appear to solely determine agent performance, there is a positive correlation between feature rank and human-normalized score. **(b)** An 'unlucky' seed from our evaluations on the sparsified version of Pong, where learning progress occurs only after the agent recovers from representation collapse.

to this discussion. Further, in all of our evaluations we find that agents whose representations have collapsed do not make learning progress. This is best exemplified by the learning curves shown in Figure 6.4b, which highlights a particularly unlucky random seed that did not observe a point being scored during its initial random exploration period and experienced representation collapse. Eventually, after several million training frames, its feature rank increases dramatically, and shortly *after* this occurs the agent solves the task. We conclude that in its extreme form, representation collapse appears to completely prevent learning progress, but that the relationship between learning progress and feature rank in less extreme cases is complex, as capacity is one of many factors influencing performance in RL. In other words: capacity is a *necessary* but not *sufficient* condition for agents to make progress.

Key insight.
Sparse-reward environments induce *feature collapse*, whereby the representation converges to a low-dimensional subspace and 'overfits' to the zero function. While it is possible to recover from feature collapse, agents fail to take successful policy improvement steps until this recovery occurs.

6.3 InFeR: mitigating capacity loss with feature regularization

The previous section showed that capacity loss can arise in online deep RL algorithms, and in some cases appears to be a bottleneck to performance. We now consider how it might be mitigated, and whether explicitly regularizing the network to preserve its initial capacity improves performance in environments where representation collapse occurs. Our approach involves a function-space perspective on regularization, encouraging networks to preserve their ability to output linear functions of their initial features. It further yields insight into the role of capacity loss in deep RL by isolating the effect of capacity on agent performance independent of the agent's exploration policy or auxiliary learning signals.

6.3.1 Feature-space regularization

Much like parameter regularization schemes seek to keep parameters close to their initial values, we wish to keep a network's ability to fit new targets close to its initial value. We motivate our approach with the intuition that a network which has preserved the ability to output functions it could represent at initialization should be better able to adapt to new targets. To this end, we will regress a set of network outputs towards the values they took at initialization. Our method, Initial Feature Regularization (InFeR), applies an ℓ_2 regularization penalty on the output-space level by regressing a set of auxiliary network output heads to match their values at initialization. Similar perspectives have been used to prevent catastrophic forgetting in continual learning [Benjamin et al., 2019], though in our case we care about the functional form of the network outputs, and not preserving the outputs associated with specific past inputs.

In our approach, illustrated in Figure 6.5, we begin with a fixed deep Q-network with parameters θ, and modify the network architecture by adding k auxiliary linear prediction heads g_i on top of the feature representation ϕ_θ. We take a snapshot of the agent's parameters at initialization θ_0, and use the outputs of the k auxiliary heads under these parameters as auxiliary prediction targets. We then compute the mean

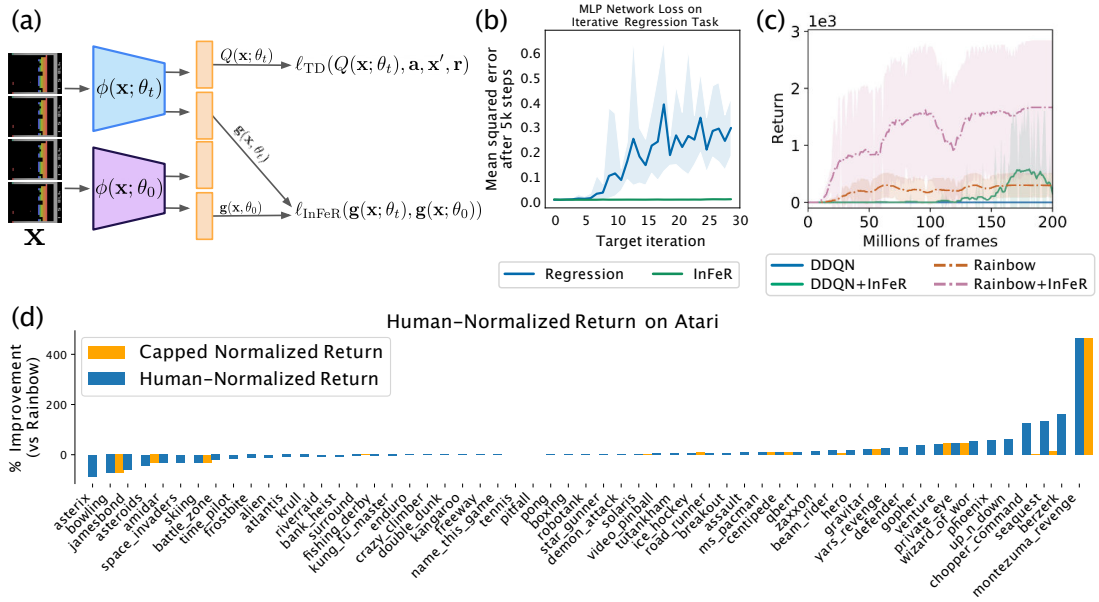


Figure 6.5: **(a)** Visualization of InFeR. **(b)** Analysis of the effect of InFeR on capacity loss. **(c)** Effect of InFeR on performance in Montezuma's Revenge with respect to Rainbow and Double DQN baselines. **(d)** Performance of InFeR relative to Rainbow on all 57 atari games.

squared error between the outputs of the heads under the current parameters $g_i(x;\theta_t)$ and their outputs at initialization $g_i(x;\theta_0)$. This approach has the interpretation of preserving subspaces of the features that were present at initialization. Because a randomly initialized network tends to produce low-magnitude outputs, a scaling factor β ensures that the regularizer target value is on a comparable scale to that of the TD targets. This results in the following form of our regularization objective, where we let $\mathcal{B}$ denote the replay buffer sampling scheme used by the agent:

$$\mathcal{L}_{\text{InFeR}}(\theta,\theta_0;\mathcal{B},\beta)=\mathbb{E}_{x\sim\mathcal{B}}\left[\sum_{i=1}^{k}(g_i(x;\theta)-\beta g_i(x;\theta_0))^2\right]. \tag{6.6}$$

The loss can then be additively combined with any RL or supervised loss function.

We first investigate whether this regularization scheme does indeed preserve capacity. To do so, we replicate the non-stationary MNIST prediction task studied in the previous section, but now incorporate InFeR. We observe in Figure 6.5 that the InFeR objective almost entirely eliminates capacity loss in a network whose performance would otherwise degrade to that of random guessing; while it is not a silver bullet, we show in Appendix D.2.3 that this effect consistently occurs across a

range of architectures and target classes. We show in Appendix D.3.2 that similar phenomena occur in RL: InFeR tends to increase the feature rank of agents trained on the Atari domain over the entire course of training; we study the early training period in Appendix D.3.3. Our findings in both settings suggest that this form of regularization may prove fruitful in a variety of continual and reinforcement learning contexts beyond the ALE benchmark studied in the remainder of this chapter.

Our analysis of RL agents trained with InFeR is made particularly insightful by the fact that InFeR does not incorporate any additional information from the environment or induce any sophisticated exploration behaviour beyond that deployed by the agent it is applied to. As a result, it allows us to isolate the effect of capacity loss on performance. If adding the InFeR objective improves performance, it can only be due to its effect on the agent's representation, or some knock-on effect thereof. We evaluate the effect of incorporating this loss in both DDQN [Van Hasselt et al., 2016] and Rainbow [Hessel et al., 2018] agents, and include the relative performance improvement obtained by the InFeR agents over Rainbow on 57 games from the Atari 2600 suite in Figure 6.5, deferring the comparison to DDQN, where the regularizer improved performance slightly on average but only yielded significant improvements on sparse-reward games, to Appendix D.3. We observe a net improvement over the Rainbow baseline by incorporating the InFeR objective, with significant improvements in games where agents struggle to obtain human performance. The evaluations in Figure 6.5 are for $k = 10$ heads with $\beta = 100$ and $\alpha = 0.1$, and we show the method's robustness to these hyperparameters in Appendix D.3.1.

The striking improvement obtained in the sparse-reward Montezuma's Revenge environment begs the question of whether such results can be replicated in other RL agents. We follow the same experimental procedure as before, but now use the DDQN agent; see Figure 6.5. We find that adding InFeR to the DDQN objective produces a similar improvement as does adding it to Rainbow, leading the DDQN agent, which only follows an extremely naive ϵ-greedy exploration strategy and obtains zero reward at all points in training, to exceed the performance of the noisy networks

approach taken by Rainbow in the last 40 million training frames. This leads to two intriguing conclusions: first, that agents which are explicitly regularized to prevent representation collapse *can* make progress in sparse reward problems without the help of good exploration strategies; and second, that this form of regularization yields significantly larger performance improvements in the presence of additional algorithm design choices that are designed to speed up learning progress.

6.3.2 Understanding how InFeR works

Having observed that our regularizer both improves performance and mitigates capacity loss, we now take a closer look into the mechanisms by which it may enable learning progress. While InFeR improves performance *on average* across the Atari games, it does not do so uniformly: its improvements are concentrated principally on games where then Rainbow agent performs significantly below the human baseline. It further clearly slows down progress in a subset of environments such as Asteroids and Jamesbond. Without a deeper analysis, it is not immediately obvious what makes games like Jamesbond and Asterix, where InFeR reduces performance, different from Seaquest and Berzerk. We now take a closer look at the mechanisms by which InFeR is shaping the agent's representation in the hopes of explaining this differential effect. We consider two hypotheses.

Hypothesis 6.3.1: InFeR improves performance by preserving a random subspace of the representation that the final linear layer can use to better predict the value function. The effect of the regularizer on other aspects of the representation learning dynamics does not influence performance.

Hypothesis 6.3.1 implies that having access to a random feature vector can improve an agent's performance even in the absence of any regularization over deeper network layers. It assumes that having a wider variety of basis functions for the final linear layer to approximate the value function will benefit the network's adaptability. To evaluate Hypothesis 6.3.1, we concatenate a single dimension of the output of a randomly initialized network to the feature outputs of the network used to learn the Q-function, and train a linear layer on top of these joint learned and

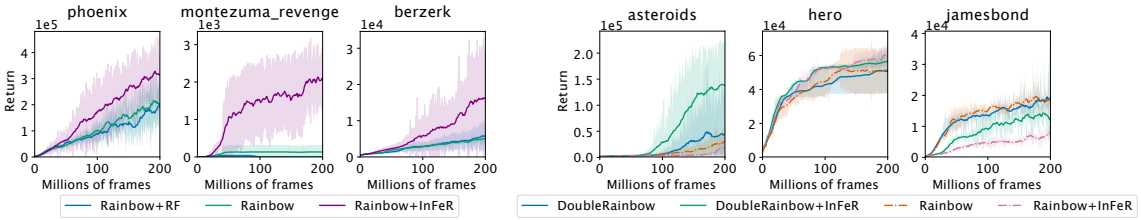


Figure 6.6: Left: agent performance does not improve over baseline when random features are added to the representation. Right: doubling the width of the neural network narrows the performance gap in games on which InFeR under-performed relative to Rainbow.

random features. We compare its performance with that of an identical network architecture with a single InFeR auxiliary head. If Hypothesis 6.3.1 were true, then we would expect this architecture to perform comparably to the InFeR agent, as the final linear layer has access to a randomly initialized feature subspace of equal dimension to that regularized by InFeR. If not, then we would expect the performance of the agents with access to the random features to be comparable to that of the vanilla Rainbow agents. Figure 6.6 shows that the latter occurs, confirming that the effect of InFeR on earlier layers is crucial to its success.

But what exactly is the effect of InFeR on earlier layers? In order to preserve its initial outputs, a network is limited in the extent to which it can modify the outputs of intermediate layers. We conjecture that InFeR limits the degrees of freedom with which a network can overfit its representation in response to early target-fitting objectives, which may also reduce the flexibility of the network to make the changes necessary to fit the current value function. In cases where representation collapse is not a concern, this will have the effect of slowing down progress. In such cases, increasing the dimension of the layer to which we apply InFeR should give the network more degrees of freedom to fit its targets, and so reduce the performance gap induced by the regularization.

Hypothesis 6.3.2: InFeR slows down the rate at which the learned features at every layer of the network can drift from their initialization in function space, regularizing the learning dynamics of the entire network. The precise subspace spanned by the auxiliary weights is not directly useful.

We test this hypothesis by doubling the width of the penultimate network layer and comparing the performance of InFeR and Rainbow on games where InFeR hurt performance in the original network. We see in Figure 6.6 that increasing the network's size reduces, eliminates, or in some cases reverses the performance gap induced by InFeR in the smaller architecture. We therefore conclude that the principal mechanism by which InFeR affects performance is by regularizing the entire network's optimization dynamics. This finding has intriguing implications on the optimal network size in reinforcement learning problems, suggesting that given suitable regularization, performance in some environments may be improved simply by increasing the network size.

6.4 Conclusions

This chapter has identified a fundamental challenge facing deep RL agents: loss of the capacity to distinguish states and represent new target functions over the course of training. We have shown that this phenomenon is particularly salient in sparse-reward settings, in some cases leading to complete collapse of the representation and preventing the agent from ever making learning progress. To address this, we proposed a novel regularizer, InFeR, with the goal of preserving capacity, yielding improved performance across a number of tasks in which deep RL agents have historically struggled to match human performance. Further investigation into this method suggests that it is performing a form of function-space regularization on the neural network, and that settings where it appears to reduce performance are actually instances of under-parameterization relative to the difficulty of the task. Particularly notable is the effect of incorporating InFeR in the hard exploration game of Montezuma's Revenge: its success here suggests that effective representation learning can allow agents to learn good policies in sparse-reward environments even under naive exploration strategies.

Our findings open up a number of exciting avenues for future work in reinforcement learning and beyond to better understand how to preserve plasticity in non-stationary prediction tasks. For example, a deeper study into the mechanisms

by which capacity is lost similar to that of Zilly et al. [2021] could provide further practically useful insights for the design of optimization algorithms and regularizers in deep learning. Of particular interest is the development of methods which preserve network plasticity. Since the submission of the paper on which this chapter is based, exciting related work has emerged proposing optimization [Dohare et al., 2021] and resetting [Nikishin et al., 2022, Zaidi et al., 2022] methods to improve network plasticity.

One mysterious finding that we have not yet fully explained is the nuanced relationship between feature rank, capacity, and generalization. We observed that feature rank and target-fitting capacity are often correlated, but that this is not always the case: in some cases, an increase in the feature rank of a network was accompanied by a decreased ability to fit 'structured' families of target functions, such as those given by the outputs of randomly initialized neural networks. Recent work has confirmed similarly nuanced relationship between feature rank and performance in offline RL [Gulcehre et al., 2022]. This observation suggests that the notion of capacity loss we discussed previously is multi-faceted: over the course of training, networks can develop inductive biases which make them better suited to fitting certain function classes than others. But why should the neural network, which is after all trained via bootstrapping on its own outputs, see a *decreased* ability to fit the outputs of other initializations of the same architecture as it improves its ability to linearly disentangle states? Chapter 7 will answer this question, bringing to the fore the question of how the learning dynamics discussed thus far come to bear on generalization in deep RL.

7 Interference and generalization

Contents

7.1 Introduction 135
7.2 Learning dynamics and smoothness 138
7.3 Generalization and interference in deep RL 147
7.4 Post-training distillation and generalization 153
7.5 Conclusions 158

7.1 Introduction

Capacity loss can be viewed as an extreme form of overfitting, where a neural network reduces not just its current performance on a set of related targets to the current training task, but also its ability to fit these targets even after many optimization steps. This chapter studies a weaker notion of overfitting, whereby training a neural network on one set of targets results in an inductive bias that is ill-suited to the value function that the network will eventually be tasked with representing. Key to this notion of overfitting is the concept of *interference*, the degree to which an update to a function approximator's output given one input influences its predictions for other input observations. We saw a form of interference presented in Chapter 4, where we discussed the degree to which a gradient update

on one minibatch reduced the loss on other minibatches in a training dataset. Here we will take a more generic view and focus on the magnitude of the change in the predicted value of one state as the result of a semi-gradient update on another. Interference can be viewed as an inductive bias encoding the similarity of the value function at different inputs. Function approximation schemes with weaker interference, such as those induced by tabular value functions or tile coding schemes, have been shown empirically to produce more stable behaviour and faster convergence in value-based algorithms on a number of classic control domains [Ghiassian et al., 2020]. However, such schemes by construction require treating the value functions for different states independently, limiting the potential for a model's predictions to generalize to new observations and resulting in *memorization*.

An array of prior empirical works demonstrate that value-based RL algorithms frequently overfit to their training environment's observations and dynamics [Lewandowski, 2020, Farebrother et al., 2018, Cobbe et al., 2021, Zhang et al., 2018c], essentially 'memorizing' the value function in a way that fails to generalize even to minor perturbations of the input. While a diverse set of training methodologies seek to mitigate overfitting [Igl et al., 2019, Raileanu et al., 2021], the source of this pathology remains under-explored. This can be attributed in part to the challenge of even defining what type of generalization is desirable in RL: unlike in supervised learning, we typically do not have access to the true optimal value function or policy in the environment, which means we cannot directly compare the performance of a policy or the accuracy of a value function to some ground truth objective [Liu et al., 2020a].

Instead, we consider interference as a proxy for generalization. While networks with a large degree of interference may not always extrapolate correctly, networks with zero interference between states will not extrapolate at all, making generalization impossible. The importance of interference to generalization has been discussed previously in Chapters 4 and 3, and was hinted at in the analysis of representation dynamics in Chapter 6, where we saw that neural networks tended to evolve representations under which states were easier to disentangle, provided that the

environment offered a sufficiently dense reward signal. We now extend this analysis to understand why this phenomenon occurs, and whether excessive disentanglement can harm generalization when the agent is exposed to new observations.

Our primary contributions in this chapter will be twofold: first, to provide a rigorous theoretical and empirical analysis of the relationship between generalization, interference, and the dynamics of temporal difference learning; second, to study the effect of distillation, which avoids the pitfalls of temporal difference learning, on generalization to novel environments. Towards this first contribution, we will extend the analysis of Chapters 5 and 6 to show that the dynamics of temporal difference learning accelerate convergence along non-smooth components of the value function first, resulting in implicit regularization towards learned representations that generalize weakly between states. Our findings present an explanation for the observation noted widely across the literature that TD-learning produces representations that are particularly vulnerable to overfitting [Raileanu and Fergus, 2021, Zhang et al., 2018c].

We then evaluate whether these findings hold empirically across a range of popular deep RL benchmarks. We measure interference by constructing a summary statistic which evaluates the extent to which optimization steps computed for one state influence predictions on other states, which we call the *update rank*. This metric is similar to the notion of feature rank proposed in Chapter 6, but applies to the updates performed by a network rather than its outputs. We find that value-based agents trained with temporal difference (TD) methods learn representations with weak interference between states, performing updates similar to those of a lookup table, whereas networks trained with policy-gradient losses learn representations for which an update on one state has a larger effect on the policy at other states. Finally, we show that post-training policy distillation is a cheap and simple approach to improve the generalization and robustness of learned policies. We find that distillation is particularly effective at increasing *smoothness* in the student network's output compared to that of the teacher. This property benefits interpolation and robustness to perturbations, however it is not a panacea:

we see that increasing smoothness via distillation does not provide significant performance improvements on generalization to new tasks that are dissimilar to the training environment. This limitation will be addressed in Chapter 8, where we will present a new representation-learning objective that targets generalization to novel environments by leveraging a notion of invariance.

7.2 Learning dynamics and smoothness

This section will explore a tension between learning dynamics in neural networks, which tend to 'generalize-then-memorize' [Kalimeris et al., 2019], and the dynamics of temporal difference learning with tabular value functions, discussed in Section 7.2.1, which tend to pick up information about the value function's global structure only late in training. We go on to study how these learning dynamics may affect the structure of gradient updates in the function approximation setting in Section 7.2.2.

Eigendecomposition of transition operators. An important concept in our theoretical analysis will be that of the eigendecomposition of the environment transition matrix. We will follow the precedent of prior work, and that of Chapter 5, in considering diagonalizable transition matrices [Machado et al., 2017, Stachenfeld et al., 2017, Mahadevan, 2005]. The relationship between the smoothness of an eigenvector and its corresponding value has been noted previously [Mahadevan and Maggioni, 2007]. However, prior discussion of this connection has defaulted to an intuitive notion of smoothness without providing an explicit definition; this intuitive notion is illustrated by the eigenvectors of the MountainCar environment visualized in Figure 7.1. We provide a concrete definition of the smoothness of a function on the state space $\mathcal{X}$ of an MDP $\mathcal{M}$ in order to provide an unambiguous characterization to which we will refer throughout this chapter.

Definition 7.2.1. *Given a function $V : \mathcal{X} \to \mathbb{R}$, MDP $\mathcal{M}$, and policy π, we define its expected variation $\nu(V)$ as follows.*

$$\nu(V) = \sum_{x \in \mathcal{X}} |V(x) - \mathbb{E}_{P^\pi(x'|x)} V(x')| \tag{7.1}$$

We say V is smooth *if $\nu(V)$ is small.*

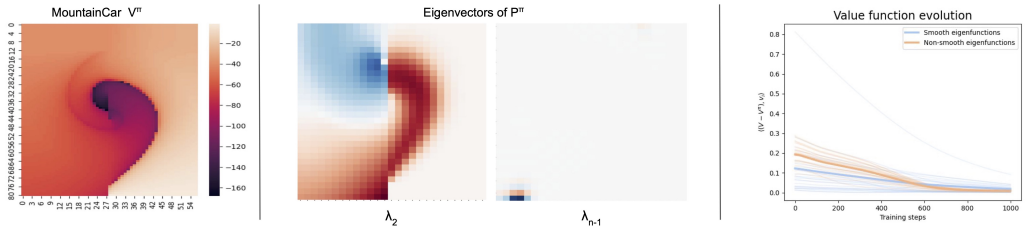


Figure 7.1: Left: value function of a near-optimal policy on MountainCar. States correspond to velocity (x-axis) and position (y-axis). Middle: eigenvectors associated with this policy computed for a discretization of the MountainCar state space. Right: value approximation error by eigen-basis coefficient along a trajectory generated by tabular TD updates with learning rate $\alpha = 0.1$ on the discretized MountainCar MDP. We compare 25 of the most-smooth eigenvectors with 25 eigenvectors corresponding to negative eigenvalues, and normalize the error by the magnitude of the projection of V^π onto the basis spanned by each set of vectors. Each transparent line corresponds to the dot product with a different eigenvector, while the solid lines show the mean over each subspace.

This expression reveals a straightforward relationship between the eigenvalue λ_i associated with a normalized eigenvector v_i and the smoothness of that eigenvector.

$$\nu(v_i) = \sum_{x \in \mathcal{X}} |v_i(x) - \mathbb{E}_{P^\pi(x'|x)} v_i(x')| = \sum_{x \in \mathcal{X}} |(1 - \lambda_i) v_i(x)| \tag{7.2}$$

In other words, the eigenvalue of an eigenvector precisely determines its smoothness. If $\lambda = 1$, for example, then the eigenvector must be constant over the MDP, whereas if $\lambda = -1$, then we have $\mathbb{E}_{P^\pi(x'|x)}[V(x')] = -V(x)$ and the expected value fluctuates between extremes when stepping from one state to another. The *variance* over next-state values can in principle be large even for functions of low variation by our definition, though in our empirical evaluations (see e.g. Figure 7.2) smooth eigenvectors tended to also exhibit little variance. For our analysis of the *expected* updates performed by TD learning, we will find the smoothness of the expected updates to be a more useful quantity than the variance.

7.2.1 Tabular dynamics

In light of this more precise definition of smoothness, we now revisit our previous results on the convergence of value functions under temporal difference and Monte Carlo learning. We recall the analysis of Chapter 5, which expressed the dynamics

of Monte Carlo (MC) updates as a continuous-time differential equation

$$\partial_t V_t = V^\pi - V_t$$

where $V_t \in \mathbb{R}^{\mathcal{X}}$ is a function on the state space $\mathcal{X}$ of the MDP, resulting in the trajectory

$$V_t = \exp(-t)(V_0 - V^\pi) + V^\pi \,.$$

Intuitively, this corresponds to a 'straight line' trajectory where the estimated value function V_t converges to V^π along the shortest path in $\mathbb{R}^{\mathcal{X}}$. In practice, most deep RL algorithms more closely resemble temporal difference updates, which are expressed as

$$\partial_t V_t = -(I - \gamma P^\pi)V_t + R_t \tag{7.3}$$

$$V_t = \exp(-t(I - \gamma P^\pi))(V_0 - V^\pi) + V^\pi \,. \tag{7.4}$$

Whereas under Monte Carlo learning the value function converges equally quickly in all dimensions, its convergence under temporal difference learning depends on the environment transition matrix. As in Chapter 5, we will consider an MDP with a diagonalizable transition operator P^π corresponding to some policy π. We write the decomposition of a function $V : \mathcal{X} \to \mathbb{R}$ as a sum of the eigen-basis vectors of P^π, written $\{v_1, \dots v_{|\mathcal{X}|}\}$, obtaining $V = \sum_{i=1}^{|\mathcal{X}|} \alpha_i v_i$ for some (unique) set of coefficients $(\alpha_i)_{i=1}^{|\mathcal{X}|}$. Under this decomposition, we can show that a predicted value function trained via TD learning will converge more slowly along smooth eigenvectors of P^π.

Observation 7.2.1. *Let P^π be diagonalizable, with eigenvectors $v_1, \dots, v_{|\mathcal{X}|}$ corresponding to eigenvalues $\lambda_1 > \cdots > \lambda_{|\mathcal{X}|}$, and let V_t be defined as in (7.4). Write $V_t = \sum_{i=1}^{|\mathcal{X}|} \alpha_i^t v_i$ to express the value function at time t with respect to the eigen-basis $\{v_i\}$. Then the convergence of V_t to the value function $V^\pi = \sum_{i=1}^{|\mathcal{X}|} \alpha_i^\pi v_i$ can be expressed as follows.*

$$\alpha_i^t - \alpha_i^\pi = \exp(-t(1 - \gamma\lambda_i))(\alpha_i^0 - \alpha_i^\pi)$$

Proof Sketch. The full proof of this result can be found in Appendix E.1.1. The key step lies in expressing the difference between V_t and V^π in terms of the eigen-basis $v_1, \ldots, v_{|\mathcal{X}|}$, for which we can obtain a closed-form expression based on the eigenvalues $\lambda_1, \ldots, \lambda_{|\mathcal{X}|}$. We consider the coefficient of V_t corresponding to the basis vector v_i, which can be written out as follows.

$$\begin{aligned}
|V_t - V^\pi|[i] &= |\alpha_i^t - \alpha_i^\pi| \\
&= |\exp(-t(1-\gamma\lambda_i))(\alpha_i^0 - \alpha_i^\pi) + \alpha_i^\pi - \alpha_i^\pi| \\
&= |\exp(-t(1-\gamma\lambda_i))(\alpha_i^0 - \alpha_i^\pi)| \\
&= \exp(-t(1-\gamma\lambda_i))|(\alpha_i^0 - \alpha_i^\pi)|
\end{aligned}$$

We note that if the basis vectors are not orthogonal, this coefficient will not be equal to the projection of V_t onto the basis vector v_i; however, understanding the evolution of coefficients still gives some insight into the convergence of smooth as opposed to non-smooth components of the value function space. □

The implications of Observation 7.2.1 on the learned value function depend to some extent on the eigendecomposition of V^π. If V^π is equal to the constant function, then we expect the high-frequency components of V_t to quickly converge to zero. If V^π puts weight on non-smooth eigenvectors, then early values of V_t may assign disproportionately large weight to these components relative to their contribution to V^π. In practice, value functions tend to exhibit a mixture of smooth and discontinuous regions. The corresponding expression of V^π with respect to the eigen-basis of P^π consequently places non-zero coefficients on eigenvectors corresponding to negative eigenvalues in order to fit this discontinuity, though its spectrum is dominated by smooth eigenvectors. We include some illustrative examples of the spectra of value functions under different assumptions on the reward and transition matrix P^π in Appendix E.3, finding in most cases that the smooth eigenvectors of the MDP tend to be assigned greater mass by the value function. The following result highlights that non-smooth components of a predicted value function, while contributing relatively little to the Monte Carlo

error, contribute disproportionately to the TD error, providing an incentive to fit these components early in training.

Theorem 7.2.1. *Let P^π be real diagonalizable with eigenvalues $\lambda_1 > \cdots \geq \lambda_n$ and $(v_k)_{k=1}^n$ the corresponding (normalized) eigenvectors. Then for any value function V, the TD error $\mathrm{TD}(V_t) = \|V_t - T^\pi V_t\|^2$ can be bounded as*

$$\|\mathrm{TD}(V_t)\|^2 = \|T^\pi V_t - V_t\|^2 \tag{7.5}$$

$$= \|\sum (1 - \gamma\lambda_i)(\alpha_i^\pi - \alpha_i^t)(v_i)\|^2 \tag{7.6}$$

$$\leq \sum_{i=1}^{n} (\alpha_i^\pi - \alpha_i^t)^2 (1 - \gamma\lambda_i)^2 \tag{7.7}$$

with equality when P^π has orthogonal eigenvectors.

The proof of this result can be found in Appendix E.1.1. Monte Carlo updates, which simply regress on the value function, give equal weight to errors along any component of the basis. These incentives provide some intuition for the different trajectories followed by Monte Carlo and TD updates: in order to minimize the TD loss, the predicted value V_t must quickly become accurate along non-smooth components of the value function V^π; however, its error due to smooth components such as the value function's bias term will have little effect on the loss and so converges more slowly. We provide an illustrative example of the relationship between the eigenvalue associated with a subspace and the convergence rate of the value function in that subspace in Figure 7.1.

7.2.2 TD learning with function approximation

Most function approximation schemes leverage the assumption that states which are close together in observation space are likely to have similar values; i.e. they encode a preference towards smooth (with respect to the observations) functions. This pushes against the tendency of temporal difference updates to encourage the learned value function to fit the components of the value function with large variation first. To investigate this tension, we consider the *kernel gradient descent* regime.

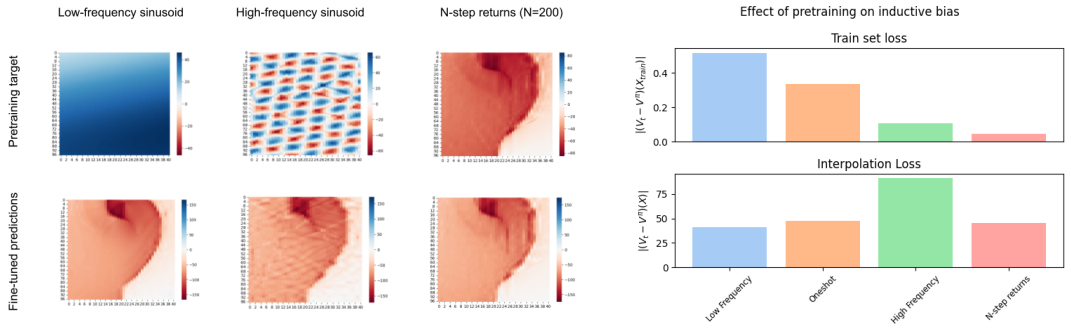


Figure 7.2: Networks trained to fit high-frequency target functions exhibit pathological interpolation properties when later fine-tuned on a value function. Left: visualization of pre-training targets (top) and final value estimate (bottom) after the pre-trained network is fine-tuned on the value function. Right: loss on the set of training states (top) and a finer-grained set of states which interpolate the training set (bottom) of each fine-tuned network.

Kernel gradient descent

Formally, a kernel is a positive definite symmetric function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. In our case, we will define $\mathcal{X}$ to be the state space of an MDP. Letting $\mathbf{x} \subseteq \mathcal{X}$, we denote by $\tilde{K}$ the (symmetric) matrix $K(\mathbf{x}, \mathbf{x})$ with entries $K(\mathbf{x}, \mathbf{x})_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$. Loosely speaking, a kernel can be thought of as measuring the similarity between two states, allowing us to encode certain forms of inductive bias into the learning dynamics of the agent. Importantly, the similarity of two states under K does not inform us about how similar the states' initial values are, but rather how an update to the value function at one state influences the value of the other; in other words, it is a proxy for the *interference* between two states. Under kernel gradient descent, the trajectory of a function is defined in terms of a kernel K and the function-space gradient of a cost function. We can translate TD semi-gradient updates into the kernel (semi-)gradient descent regime as follows, where we let R^π denote the expected reward under policy π.

$$\partial_t V_t = \tilde{K}((\gamma P^\pi - I)V_t + R^\pi) \tag{7.8}$$

It is straightforward then to obtain analogous results as before on the convergence of V_t to V^π based on the eigendecomposition of the matrix $\tilde{K}(\gamma P^\pi - I)$ in cases where this matrix is positive definite, though in general it may have negative

eigenvalues. This decomposition will not in general have a closed form in terms of the eigendecompositions of $\tilde{K}$ and P^π, but special cases have been studied in the setting of linear regression by Ghosh and Bellemare [2020] and can be related to kernel gradient descent straightforwardly as discussed in Appendix E.1.1. This setting also describes the dynamics of neural networks in the limit of infinite width [Jacot et al., 2018, Fort et al., 2020, Lee et al., 2020], which follow kernel gradient descent with respect to the neural tangent kernel.

A more interesting case occurs when we assume some states in the environment are not updated during training. In this case, we can characterize their evolution (and potential influence on the training set via bootstrapping) using the direct sum of $K(X_{\text{train}}, X_{\text{train}})$ and $K(X_{\text{test}}, X_{\text{train}})$.

Theorem 7.2.2. *Let K be a kernel and π a fixed policy in an MDP with finite state space X. Let $X_{\text{train}} \subset \mathcal{X}$ be the set of states in the support of π, $X_{\text{test}} = \mathcal{X} \setminus X_{\text{train}}$, and let V_t be a value trajectory obtained by applying kernel semi-gradient updates to some initial value function V_0 with kernel K. Let K_{all} be defined as*

$$K_{\text{all}} = K(X_{\text{train}}, X_{\text{train}}) \oplus K(X_{\text{test}}, X_{\text{train}}) \tag{7.9}$$

Then the trajectory of V_t on the entire state space X will be as follows,

$$\partial_t V_t(X) = (K_{\text{all}})[(T^\pi V_t - V_t)(X_{\text{train}})] \,. \tag{7.10}$$

A full derivation is provided in Appendix E.1.1. These dynamics diverge notably from the standard kernel gradient descent regime in that changes to predictions on the test set have the potential to influence the dynamics of V_t on the training set. This situation can arise when the agent is trained on a collection of sampled transitions which do not correspond to a contiguous trajectory, for example if transitions are removed from a replay buffer via some prioritization scheme, rather than in order of recency. In off-policy learning it can also arise when the policy used to construct the bootstrap target differs from that used to collect the replay buffer data. A large value of $K(X_{\text{test}}, X_{\text{train}})$ implies that updates to the training set hold influence over predictions on the test set, but at the cost of increasing asymmetry in

K_{all} when viewed as an operator on $\mathbb{R}^{\mathcal{X}}$. In Appendix E.3.2 we illustrate how this asymmetry can harm stability in the case of a simple radial basis function kernel. This can be viewed as a special case of off-policy temporal difference methods where the probability of sampling some states is set to zero [Ghosh and Bellemare, 2020].

Combining insights from Theorem 7.2.2 and Observation 7.2.1, we arrive at an intriguing conclusion: in the case of smooth kernels, the components of the value function most suitable to approximation via the kernel K are precisely those which appear in the value estimate of the training set only later in the trajectory. As a result, the kernel does not receive the necessary information to generalize accurately to new observations until late in the training process. This observation runs contrary to the standard kernel regression regime, where one argument in support of early stopping is that kernel gradient descent methods converge along smooth components fastest [Jacot et al., 2018]. At the same time it is an obvious effect of bootstrapping, which requires that the agent update its predictions several times in order to propagate information about the value function through the entire input space. This effect is illustrated in Figure E.4 in Appendix E.3.2.

Non-linear function approximation

We saw in Chapter 6 that the evolution of a network's representation is crucial to its ability to adapt to new learning signals. We now turn our attention to how the evolution of a network's features can influence its generalization properties. Our primary object of focus will be the gradient structure of a function approximator, whose analysis depends on the second-order effects of TD semi-gradient updates under finite step sizes. We consider the system

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_\theta V(\theta_t) \cdot [(\gamma P^\pi - I)V(\theta_t) + r] \tag{7.11}$$

which can be viewed as an Euler discretization of the dynamics described in (7.4). We will use the notation $f(\theta_t)$ to refer to the semi-gradient update on parameters θ_t inducing value function V_{θ_t}, and write $\text{TD}(\theta) = \frac{1}{2}\|V_\theta - \square T^\pi V_\theta\|^2$, where the $\square$ denotes a stop-gradient. Using the continuous time system in (7.4) to approximate

these updates for time t will gradually accumulate increasing errors, proportional to $(\alpha n)^2$, as it does not take into account the effect of the discrete step size on higher-order gradients of V_θ. We apply a similar analysis to that of Barrett and Dherin [2021] and Smith et al. [2020] to understand the effect of the discrete learning dynamics on the gradient structure of V_θ itself. We let

$$f_1(\theta) = -\frac{1}{2}\nabla_\theta \|\nabla_\theta \mathrm{TD}(\theta)\|^2 + \gamma(\nabla_\theta^\top V P^\pi \nabla_\theta V) f(\theta) \tag{7.12}$$

to obtain a second-order correction describing the effect of gradient descent on the gradient structure of the learned representation.

Observation 7.2.2 (Second-order dynamics)**.** *Let θ_t be defined by the discrete-time system (7.11) with step size α. Let $f_1(\theta)$ be defined as in (7.12). Let $\tilde{\theta}_t$ denote the trajectory obtained by following the dynamics:*

$$\partial_t \tilde{\theta}_t = f(\tilde{\theta}_t) + \frac{\alpha}{2} f_1(\tilde{\theta}_t)\ . \tag{7.13}$$

Then we have $\theta_n \approx \tilde{\theta}_{n\alpha} + O((n\alpha)^3)$, where $\tilde{\theta}_{n\alpha}$ denotes the value of $\tilde{\theta}_t$ at time $t = n\alpha$.

The proof of this result follows a similar structure as that of Smith et al. [2020] and can be found in Appendix E.1.1; the key difference from the gradient descent regime is that we must now take into account the influence of gradient updates on the loss via the Bellman targets. This distinction presents itself in the form of f_1 constructed in (7.12): the first term in the sum consists of a semi-gradient norm penalty term with respect to the instantaneous TD error, while the second term partially offsets this penalty along smooth components of the MDP, accounting for the target drift due to gradient updates. The first term is the standard gradient norm penalty observed in supervised regression. The second term has been studied in the setting of offline RL by Kumar et al. [2022], where it plays a greater role than in the online RL problems in this chapter.

The effect of this penalty term on the evolution of interference in deep neural networks is difficult to analyze in closed form due to the influence of the network Hessian, however our empirical observations from Chapter 6 provide an illustrative example. We recall that the previous section showed that early TD targets will

typically be less smooth than the true value function. The gradient norm penalty will thus implicitly discourage interference between states in order to fit this relatively discontinuous function in a manner robust to noise in the optimization process. This observation is foreshadowed in the findings of Chapter 6, which found that deep RL agents learn to map nearby states to distinct feature vectors in dense-reward environments. We illustrate an analogous effect on interference in Figure 7.4, where networks trained in dense-reward games (whose early TD targets will exhibit greater variation) exhibit weaker interference after training than they did at initialization. We can further observe the long-term impact of fitting non-smooth targets on the inductive bias of a network in Figure 7.2, where networks trained to fit high-frequency sinusoid functions exhibit pathological interpolation behaviour when later fine-tuned on the target value function.

In combination, the findings of this section suggest that the dynamics of temporal difference learning work to discourage interference between states in deep RL by fitting high-frequency components of the value function early in training while also encouraging robustness of the loss to noisy optimization steps. While this may result in more stable learning, as highlighted in Theorem 7.2.2, it has the double-edged effect of reducing the degree to which the network may generalize to novel observations. We now leverage these results to gain insight into deep RL agents trained in rich-observation environments.

7.3 Generalization and interference in deep RL

The inductive bias of a neural network is influenced heavily by the initial training period [Frankle et al., 2020b, Achille et al., 2018, Golatkar et al., 2019]. Much like in human development, the data a neural network sees in the early stages of optimization sets the network's inductive bias in a way that can be difficult to unlearn later. The distribution of states and the targets that the agent is trained to fit during this period therefore have an outsize influence on what types of targets the network will be able to easily fit down the road. In Chapter 6, we saw that networks which did not receive any reward signal early in training had greater

difficulty distinguishing states later in training. We now dive deeper into how dense rewards shape agents' representations: whereas sparse rewards discourage state disentanglement, we conjecture that dense reward environments may encourage excessive disentanglement – in other words, *discourage* interference – at the expense of generalization in value-based deep RL algorithms. We begin by presenting a quantitative approach to measure the degree to which interference is occurring between states in the agent's visitation distribution. Armed with this metric, we evaluate two concrete hypotheses. First, that deep neural networks trained with TD updates will exhibit weaker interference between states as training progresses compared to their value at initialization (Hypothesis 7.3.1). Second, that networks trained with TD learning will exhibit weaker interference than those trained with policy gradient objectives (Hypothesis 7.4.2).

7.3.1 Representation evolution in DQN agents

We begin by presenting a method to track interference in deep RL agents. Given a set of transitions $\tau_1, \ldots, \tau_n$ of the form $\tau_i = (\mathbf{x}_i, a_i, r_i, \mathbf{x}'_i)$ and a value function V with parameters θ, we let θ_i denote the network parameters after performing an optimization step with respect to the transition τ_i. We then construct a matrix $\mathbf{I}_\Delta$ entry-wise by computing the interference $\mathrm{I}_\Delta(\mathbf{x}_i, \mathbf{x}_j)$ between each pair of states $\mathbf{x}_i, \mathbf{x}_j$ in the sampled set. We refer to this object as the *update matrix*, as it consists of the changes in the network outputs as a result of gradient updates. Recalling (2.14), we let $Q_\theta(\mathbf{x})$ denote the vectorized action-value function output by a DQN network at state $\mathbf{x}$ given parameters θ, g_ℓ the TD semi-gradient update direction, and η the optimizer state.

$$[\mathbf{I}_\Delta]_{i,j} = \mathrm{I}_\Delta(\mathbf{x}_i, \mathbf{x}_j) = \|Q_\theta(\mathbf{x}_j, \cdot) - Q_{\theta'}(\mathbf{x}_j, \cdot)\| \text{ where } \theta' = \theta + \alpha g_\ell(\mathbf{x}_i; \theta, \eta) \quad (7.14)$$

Figure 7.3 provides an illustration of the update matrix of a radial basis function kernel regression model where the lengthscale of the kernel is set to different values. In DNNs, the properties of this matrix will depend on the optimizer used to perform updates, leading to notable differences from the neural tangent kernel regime studied

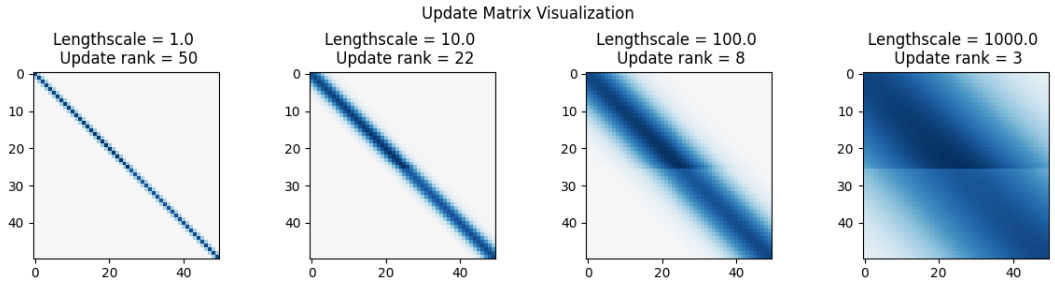


Figure 7.3: Update matrices for a toy kernel regression problem using RBF kernels of different lengthscales. The smaller lengthscales discourage generalization, and induce update matrices with a more pronounced diagonal component.

elsewhere [Yang et al., 2022] in the case of non-linear function approximators trained with adaptive optimizers. At one extreme, the update matrix $\mathbf{I}_\Delta$ for a tabular value function (demonstrated by the short lengthscale kernel plot on the left hand size of Figure 7.3) will have non-zero entries only along the diagonal and the matrix will have full rank. At the other, if the value function is represented by a single parameter $\theta \in \mathbb{R}$, then every row will be identical up to a scalar multiple and the matrix will have rank one. More generally, the rank of this matrix can be interpreted as a proxy for whether an agent tends to *generalize* updates between states (low rank), or whether it *memorizes* the value of each state-action pair independently from other states (high rank). As in Chapters 3 and 6, we use an approximate version of the rank that discards negligible components of the matrix based on the singular value decomposition, analogous to the feature rank quantity computed in the previous chapter. We provide full computation details in Appendix E.2.1

$$\rho(\mathbf{I}_\Delta) = |\{\sigma | \sigma \in \text{SVD}(\mathbf{I}_\Delta) \text{ and } \sigma > \epsilon\}| \tag{7.15}$$

We will refer to $\rho(\mathbf{I}_\Delta)$ as the *update rank*. Typically, $\mathbf{I}_\Delta$ will depend on the optimization state θ (which describes the network architecture, parameters, and optimizer state) of the agent and so we will sometimes write $\mathbf{I}_\Delta(\theta)$ to denote this dependence when it is not obvious. An alternative approach outlined by Daneshmand et al. [2021] involves computing the Frobenius norm of the difference between the matrix $\mathbf{I}_\Delta$ and the identity, however this may overestimate interference in optimizers which use momentum due to constant terms in the update matrix. In

our case, a change in the rank of $\mathbf{I}_\Delta(\theta_t)$ and $\mathbf{I}_\Delta(\theta_{t+n})$ after n steps of optimization indicates that the inductive bias of the network has shifted to generalize more, in the case of a decrease in rank, or less, in the case of an increase in rank, between states. The update rank thus gives us a means of testing our first empirical hypothesis of this chapter.

Hypothesis 7.3.1: deep neural networks trained with TD updates exhibit weaker interference between states as training progresses.

We measure interference by evaluating $\rho(\mathbf{I}_\Delta(\theta))$ for states sampled from the replay buffer of a deep RL agent. Hypothesis 7.3.1 predicts that the rank $\rho(\mathbf{I}_\Delta(\theta_t))$ will increase as a function of the training step t. To test this we train a standard DQN architecture on environments from the Atari 2600 suite, and save a range of checkpoints throughout training. We illustrate the evolution of agents' update matrices $\mathbf{I}_\Delta(\theta_t)$ over the course of training in Figure 7.4. We observe that RL agents trained in dense-reward environments tend to develop update matrices which resemble those of tabular value functions: they tend to have a pronounced diagonal or block-diagonal structure later in training, compared to a more linear structure early in training. Those trained in the absence of reward, i.e. those for which the target value function has no high-frequency components, maintain low-rank update matrices through training as our theory would predict. We find that similar results hold for a range of update rules, including distributional updates performed in the C51 algorithm [Bellemare et al., 2017]. Quantitatively, we track the update rank of these checkpoints on the right hand side of Figure 7.4. We include further evaluations in Appendix E.4.

7.3.2 Actor-critic methods

Hypothesis 7.3.1 was motivated by the observation that early temporal difference targets predominantly contain information about immediate rewards, implicitly rewarding networks which memorize rather than generalize the learned value function. However, policy gradient methods do not encounter this issue: these methods seek

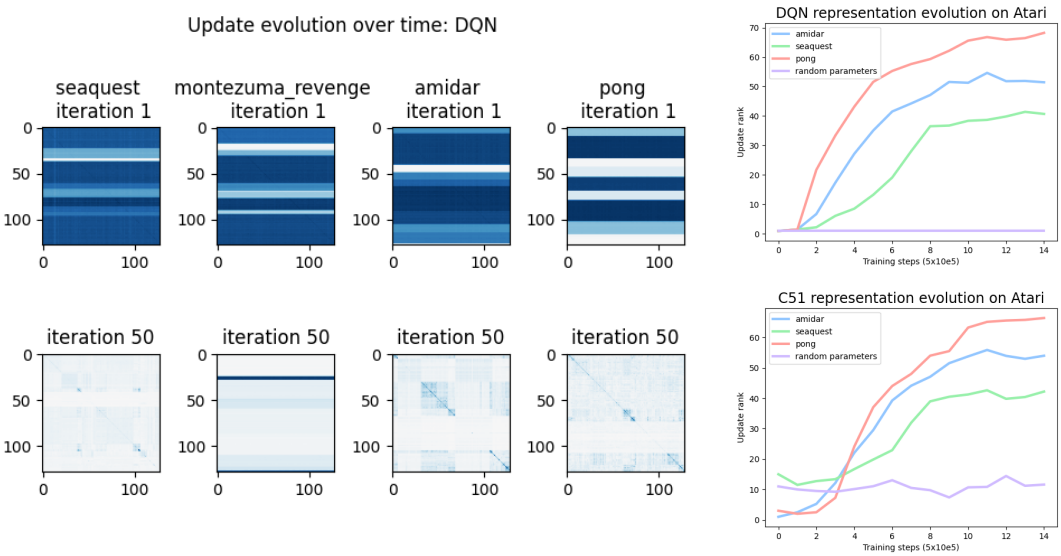


Figure 7.4: Left: $\mathbf{I}_\Delta(\theta_t)$ for agents trained on games from Atari. The networks initially exhibit low update rank, but after 50 iterations (5M frames of experience), the updates rank increases significantly. This is tracked in the right hand side plots over the course of approximately 7M frames. Random parameters refers to the update rank obtained by a randomly initialized neural network.

to maximize the probability of selecting an action with large advantage, which is independent of the smoothness of the TD targets. While policy gradient methods encounter their own challenges, in particular high gradient variance, these challenges are largely orthogonal to the particularly adversarial nonstationarity of TD learning. Actor-critic methods, which involve training a policy via policy gradient methods and a value function via temporal difference methods, therefore present an ideal test bed to study whether the decline in generalization observed during training in the previous section is unique to TD updates, or whether it is a more general property of RL agents. A priori, we have no reason to expect the policy gradient losses to exhibit significant increases in structure later in training. We therefore propose the following hypothesis.

Hypothesis 7.3.2: networks trained with TD learning will exhibit weaker interference than those trained with policy gradient objectives.

We repeat the analysis of the previous section with actor-critic methods to study whether this is the case, measuring the change in the actor's output policy at a

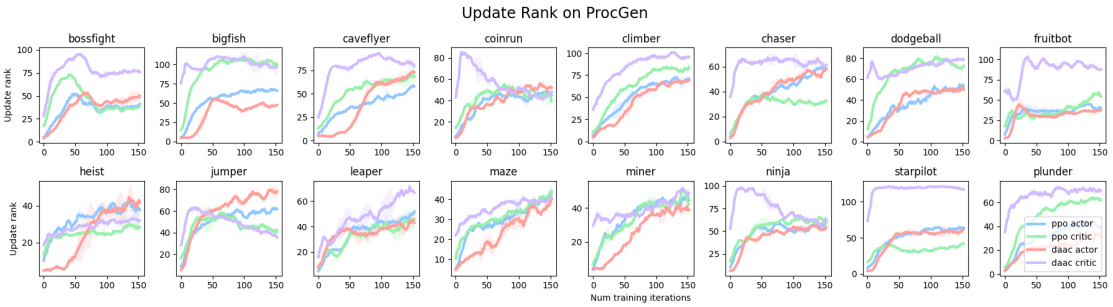


Figure 7.5: Update dimension of actor-critic methods in ProcGen. Shading indicates minimum and maximum values over 4 seeds. We observe that the update dimension of the separate critic architecture in DAAC (the lilac line) consistently has the highest update rank early in training, while the actors have the lowest rank in the early training stages and only surpass the DAAC critic later in training.

state $\mathbf{x}$ rather than Q-values. We run our evaluations in the ProcGen environment [Cobbe et al., 2019], which consists of 16 games with procedurally generated levels. While the underlying mechanics of each game remain constant across the different levels, the layout of the environment may vary. The agent is given access to a limited subset of the levels during training, in this case 10, and evaluated on novel randomly generated levels. In this section we study interference only on the training environments. We investigate Hypothesis 7.4.2 using two different algorithms on the ProcGen suite: PPO [Schulman et al., 2017], which uses a shared representation network for both the actor and critic, and DAAC [Raileanu and Fergus, 2021], where there are no shared parameters between the actor and the critic. We then evaluate the update dimension of the two different methods, and plot the results in Figure 7.5. Additional details can be found in Appendix E.2.2. Omitting the critic gradients from the actor's representation leads to significantly lower update dimensions early in training in a number of environments, including bigfish, heist, and miner. Further, the critic network in DAAC, which receives only TD gradients, exhibits markedly higher update rank in all environments in at least the early stages of training, and often throughout the entire trajectory, than the other networks which have access to the actor gradients.

7.4 Post-training distillation and generalization

The previous section has demonstrated that training value-based deep RL agents results in a bias towards weaker interference between inputs. Arguably, this weak interference may be beneficial to the stability of these learning algorithms, but it comes at the cost of generalization. This bias towards memorization arises when, during the network's crucial early development stage, it is trained to fit target functions that do not capture the global structure of the value function. This leaves us with an open question: how might we adapt the training procedures followed in deep RL to obtain agents that can adapt to inputs they have not seen before?

One simple solution to this problem is to train a freshly initialized network on the final value function obtained by TD learning. If the teacher network was able to fit the low-frequency components of the value function – even if this was achieved via memorization – then the freshly initialized network will be able to benefit from incorporating this structure into its predictions from the start of its optimization procedure. In the case of policy distillation in actor-critic algorithms, it allows us to evade the influence of critic gradients without requiring a complete decoupling of the actor and critic during training time. Such approaches have seen success in prior work [Igl et al., 2019, Nikishin et al., 2022]; this section presents a deeper study of a mechanism driving this success.

7.4.1 Value distillation

We begin by studying the effect of post-training distillation on robustness of the learned value function in environments from the Atari suite [Bellemare et al., 2013]. We first consider value distillation as a means of eliminating the counterproductive bias towards memorization induced by early TD targets. We leverage a data collection policy from a pre-trained teacher network q_t, and perform distillation of a freshly initialized network q_s on this data. We follow a similar procedure to that of Ostrovski et al. [2021] to perform distillation of the function q_s on data collected sampled from the teacher's replay buffer $\mathcal{B}_T$, leveraging their insight that distillation on *all* action values, rather than only the value of the action taken by

the teacher agent, yields significantly higher performance. We additionally study the effect of behaviour cloning with entropy regularization, obtaining the objectives

$$\ell_{\mathrm{VD}}(q_{\mathrm{S}}, q_{\mathrm{T}}) = \mathbb{E}_{s\sim\mathcal{B}_{\mathrm{T}}}\left[\sum_{a\in\mathcal{A}}(q_{\mathrm{S}}(a) - q_{\mathrm{T}}(a))^2\right] \tag{7.16}$$

and

$$\ell_{\mathrm{BC}}(\theta) = \mathbb{E}_{s,a\sim\mathcal{B}_{\mathrm{T}}}[\log \pi_\theta(s,a) + \lambda H(\pi_\theta(s))] \tag{7.17}$$

where $H(\cdot)$ denotes the entropy of the policy. We set $\lambda = 0.01$ in our evaluations. We show results for value distillation (7.16), which regresses on the outputs of the frozen Q-network, and behaviour cloning (7.17), which predicts the action taken by the frozen Q-network. We track three quantities: the performance of the learned policy, the robustness of the learned policy to perturbations, and the consistency of the learned policy when interpolating between observations. The performance is measured by following an ϵ-greedy policy in the training environment, with $\epsilon = 0.01$. The robustness to perturbations is measured by tracking whether the network takes the same action under a Gaussian perturbation to its input as in the unperturbed observation. Finally, we investigate the network's interpolation behaviour by evaluating whether, given a convex combination of observations o_1 and o_2, the network takes the same action under the combination as it does in either of the original observations. Additional evaluation details can be found in Appendix E.2.3. We are interested in evaluating the following hypothesis.

Hypothesis 7.4.1: post-training distillation will reduce overfitting and therefore improve robustness; this will hold more strongly for policy distillation than for value distillation due to the discontinuity of the value distillation targets.

Figure 7.6 shows that the distillation approaches yield policies that exhibit significant improvements in robustness to perturbations and are more consistent under interpolations between observations. We observe that the behaviour cloning method matches or nearly matches the performance of the pretrained agent in three of the four environments, while also obtaining the best robustness. Both behaviour

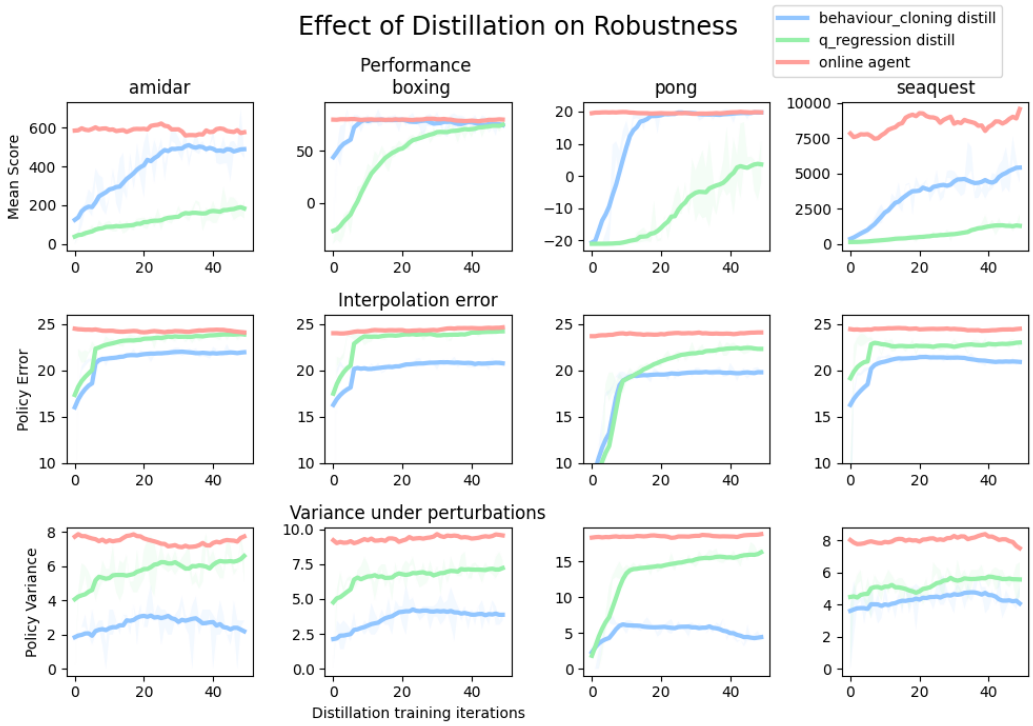


Figure 7.6: Performance and robustness to perturbations of different distillation approaches in games from the Atari suite. Post-training distillation results in policies that are more consistent under perturbations and under interpolation between observations. Axes indicate the ℓ_1 norm between the policy on the original input batch and on the perturbed input batch.

cloning and value distillation improve upon the robustness of the teacher network that was trained online. We conclude that both value and in particular policy distillation increase smoothness in the form of improved robustness to perturbations and greater interpolation consistency. This finding motivates the next section, where we will dig deeper into policy distillation.

7.4.2 Policy distillation

We have previously shown that decoupling the policy and value function approximators in an actor-critic architecture resulted in weaker interference in the value approximator, and stronger interference in the policy network. However, it is not clear that this should necessarily lead to better final performance on the test environment, as the interaction between the policy and value function training is complex and difficult to predict – a policy with a small generalization gap but weak training performance may nonetheless perform worse at evaluation time than

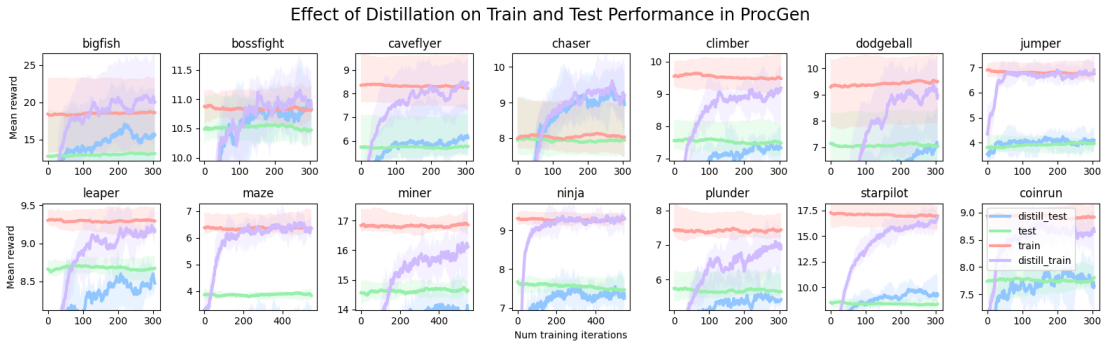


Figure 7.7: Effect of policy distillation on generalization in environments from the Procgen suite. We plot the pretrained networks train environment and test environment performance, along with the performance of the distilled agent on the test environments. We see significant improvement on test environments in bigfish, caveflyer, chaser, climber, and bossfight.

a policy with a larger generalization gap. An alternate approach to remove the influence of critic gradients on the learned policy is to train a behaviour cloning agent on the final policy obtained after training. Because we only need to distill the policy, we avoid contaminating the actor with value approximation gradients while (hopefully) preserving the performance of the original agent on the training environments. We return to the ProcGen benchmark, with the hypothesis that post-training distillation of PPO agents should produce policies which improve on the ability of the final trained actor to generalize to new levels.

Hypothesis 7.4.2: post-training policy distillation will reduce the generalization gap of actor-critic agents trained on procedurally generated environments.

Hypothesis 7.4.2 brings with it a subtle but crucial caveat: our discussion of generalization thus far has focused implicitly on *within-environment* generalization, but procedurally generated benchmarks inherently evaluate agents on novel environments, inducing a distribution shift. Our primary tool in studying generalization in the previous section, the update rank, carries with it the assumption that a network's generalization behaviour on training observations will be representative of its behaviour on novel states that may be encountered later. However, the ProcGen benchmark requires generalization *out-of-distribution* to novel environments whose observations may have probability zero under the training environments. A neural

network will not necessarily exhibit similar generalization behaviour in a novel environment which may be highly visually dissimilar to its training environment. As such, methods such as policy distillation which increase generalization between observations in the training environment should only be expected to benefit test set performance if the test environments are sufficiently 'within-distribution'.

With this caveat in mind, we proceed to evaluate Hypothesis 7.4.2. We train a set of PPO agents on the difficult setting of the ProcGen benchmark, using the same experiment configurations as that used by Cobbe et al. [2020]. We then perform behaviour cloning on these agents, using data collected only from the training environments, by training a freshly initialized network (the distillation agent) to minimize an entropy-regularized KL divergence with the teacher's policy on trajectories collected by the teacher. Training was terminated at 50M sampled environment steps, except for Maze, Miner, and Ninja, which were run for 90M sampled steps as the training performance of the student had not yet matched that of the teacher after the initial budget. We then evaluate the distilled agent's performance on the test environments. Results are shown in Figure 7.7.

We find a rich heterogeneity in the shape of the learning curves of the student on the train and test environments across the different ProcGen games. Post-training distillation in most settings slightly reduces the generalization gap obtained by the original network, with the notable exception of the 'maze' environment. In settings where the gap between train and test performance is larger, we see a more mixed effect from distillation. The naivety of our approach suggests that it is likely that final performance and sample-efficiency of these agents could be improved using tools from the policy distillation literature such as allowing the student to generate training data [Czarnecki et al., 2019, Rusu et al., 2016, Teh et al., 2017].

We observe three broad categories of behaviour in the student agent's learning curves. First, in the two environments where PPO achieves a negligible generalization gap, distillation improves not only test performance but also performance on the training environments. This suggests that the memorization behaviour of PPO was limiting not only its test set performance but also its returns on training

environments. We further do not observe any evidence of stereotypical overfitting in these environments: the train and test performance increase in lockstep with each other. In environments where there is a moderate gap between train and test environment teacher performance, the test set performance of the student tends to increase more slowly than its training set performance, resulting in slow but fairly consistent convergence of the learned policy to approximately the level of that of the teacher (or slightly exceeding that in the case of bigfish, caveflyer, jumper, and starpilot). In no environments do we see the classic sign of overfitting: a sequence of increasing, saturating, and decreasing training performance. Instead, the final test performance depends more on the relative positive slope of the test and the training performance curves – in other words, it depends on the degree to which the agent is able to identify shared structure between the training environments that is more sophisticated than can be detected by measuring extrapolation between observations within a single training environment. This notion of shared structure will be explored in greater depth in Chapter 8.

7.5 Conclusions

The analysis of this chapter presents a complementary perspective to that of Chapter 6 in studying how learning dynamics can influence generalization, particularly in dense-reward environments. Our key take-away is that the nature of the targets that we ask a neural network to fit early in its training trajectory hold significant influence over the bias of the learning process towards certain types of functions later. In the case of sparse-reward environments, networks which overfit to the task of predicting the zero target struggled to fit any non-trivial target function later in training. In the case of dense rewards, we've shown in this chapter that networks evolve to be more prone to memorization as a result of fitting discontinuous and unstructured targets early in training. Broadly, our analysis suggests that the nature of the learning dynamics of value-based RL discourages generalization in deep neural networks. We have shown that temporal difference learning in the continuous-time dynamics setting of Chapter 5 fits non-smooth components of the value function first, resulting

in an implicit bias towards representations that encode near-tabular updates. In the context of prior work demonstrating that weaker generalization can improve the stability and convergence of RL algorithms, this phenomenon may be beneficial to an agent's stability at the cost of observational overfitting. We further show that post-training distillation improves generalization and robustness, mitigating some of the tendency of value-based RL objectives to encourage overfitting.

The findings presented here are a crucial stepping stone along the path to the principled development of robust and stable deep RL algorithms which are capable of strong generalization performance. Our insights may prove useful in a range of future directions, such as using different architectures during training and distillation, leveraging larger neural network function approximators to minimize harmful interference, and modifying the update rule used in TD learning to adaptively promote or inhibit interference between inputs. Further, the role of the optimizer is fundamental to the phenomena studied in this paper, and RL-specific optimization approaches may benefit from our findings. The notion of update rank that we consider here may also be adapted to other settings to shine a light onto where a network lies on the memorization-generalization spectrum at different points during training, potentially providing a mechanism to trigger early stopping in supervised learning settings, or as a meta-learning objective for hyperparameter tuning and architecture search.

The experiments presented in this chapter have considered both single-environment (in the case of Atari and MountainCar) and multi-environment (ProcGen) problems. While our findings concerning interpolation and robustness to perturbations in single-environment RL are straightforward, analysis of the multi-environment setting is more subtle. The update rank we compute on agents measures the degree of generalization between observations from the training environments. Because the test environments may differ systematically from the training environments, the update rank may not capture properties of a neural network that enable effective generalization to these out-of-distribution test examples. Indeed, based on our observations from Chapter 4 we have no reason to expect that invariance over training

inputs will generalize to sufficiently out-of-distribution test environment observations. The following chapter will study the multi-environment problem explicitly. It will outline sufficient assumptions on the structure of the data-generating process to ensure generalization is tractable, and present a novel representation-learning approach that is able to identify the shared structure between the training and test environments.

8

Generalization across environments

Contents

8.1 Introduction 161
8.2 Background on causality 164
8.3 Problem setting 166
8.4 State abstractions and causal feature sets 172
8.5 State abstractions for rich observations 179
8.6 Conclusions 188

8.1 Introduction

A robot's LiDAR sensor is knocked askew. An autonomous vehicle encounters hail for the first time. In the distant future, a RoboChef discovers that the kitchen in which it usually works has been remodelled. Reinforcement learning agents will frequently encounter situations at deployment that they did not see during training. Such situations will not necessarily correspond to novel observations from the agent's training environments, but they may share a broader notion of *causal structure*. An agent which has learned to robustly exploit this structure will generalize with ease to the challenges it faces at deployment. However, in such cases good generalization between observations from the training environment will not necessarily be sufficient for an agent to pick up on this causal structure. We saw the limitations of such

generalization in Chapter 7 on the ProcGen benchmark. Reducing the degree of memorization performed by a neural network improved some types of robustness, but did not always result in better performance on test environments. In the worst case, some training environments may contain spurious correlations that will not be present at test time. An agent which depends on these correlations may then generalize well to new observations in its training environments, but experience catastrophic failure on new environments where the correlation is not present [Zhang et al., 2018b, Song et al., 2020]. Two things are clearly necessary in order to obtain effective generalization outside of the training environment. First, we must look beyond interference to more sophisticated notions of invariance. Second, we must consider families of test environments for which the training environments provide sufficient information for generalization to be possible.

Generalization to new environments has been a topic of great interest to the RL community, but this interest has been concentrated on settings that make few explicit assumptions on the shared structure between training and test environments [Kirk et al., 2021]. Recent works [Amit and Meir, 2018, Yin et al., 2019] have developed generalization bounds for the multi-task problem, but they depend on the number of tasks seen at training time, which can be prohibitively expensive given the sample complexity of RL even in the single task regime. To obtain stronger generalization results, we consider a multi-environment RL problem: like multi-task RL, the agent seeks to maximize return on a set of environments, only some of which are available to the agent during training. We make the assumption that there exists some latent causal structure in the form of a causal graph that is shared among all of the environments, and that the sources of variability between environments can be modelled by interventions on spurious variables in the causal graph. This family of environments, which we show to be equivalent to a *Block MDP* [Du et al., 2019], allows for observations to vary, but fixes the latent states, dynamics, and reward function.

Chapter 7 studied multi-environment generalization, but with relatively few assumptions on the shared structure between the different environments; we only

required that the environments be drawn from the same data-generating process. We show in this chapter that the added assumption of shared structure allows for much stronger generalization results than have been obtained by prior work with significant sample complexity improvements. Our results are enabled by a key insight: we turn the problem of identifying features that will generalize well to novel environments from a learning problem, for which sample complexity may be prohibitive, to a causal identification problem, which will often be more tractable. Indeed, where Cobbe et al. [2019] and Zhang et al. [2018a] find that agents trained using standard methods must see many thousands of training environments in order to successfully generalize to new environments, in our more restricted setting as few as two or three training environments can be sufficient to identify the correct causal structure.

The main technical contribution of this chapter is the application of tools from causal inference to identify (and in some cases learn) representations that will enable an agent to generalize well to new environments. We propose a method motivated by a simple intuition: features which remain invariant across the different training environments are likely to be causally related to the reward and transition structure. Policies which depend on these features will therefore be robust to changes in the environment brought about by interventions on spurious state variables. In certain linear function approximation settings, we demonstrate that this method will, with high probability, learn an optimal state abstraction that generalizes correctly to novel environments, requiring many fewer training environments than would be necessary without the block MDP assumption. We then draw a connection between bisimulation and the minimal causal set of variables found by our algorithm, providing bounds on the model error and sample complexity of the method. We further show that using analogous invariant prediction methods for the nonlinear function approximation setting can yield improved generalization performance over multi-task and single-task baselines.

8.2 Background on causality

A desirable property of a learned representation is that it should enable a policy trained on one environment to generalize well to new environments. A promising approach to learn these general representations, particularly in the partially observable setting, comes from training an agent to learn the causal structure of the environment [Zhang et al., 2019, de Haan et al., 2019]. We refer to Section 2.3.1 for a discussion of state abstractions. This section will provide the relevant background on causality; the two concepts will be related in Section 8.4. Causal inference [Pearl, 2000] is a powerful tool that machine learning researchers are beginning to leverage [Johansson et al., 2016, Louizos et al., 2017], developing explicit connections to generalization bounds and robustness [Shalit et al., 2017]. While a great deal of work on causality focuses on learning an explicit causal graph [de Haan et al., 2019], an alternative approach involving invariant prediction [Magliacane et al., 2018, Peters et al., 2016, Rojas-Carulla et al., 2018] is in fact closely aligned with the literature on generalization bounds discussed in Chapter 2.

8.2.1 Causal discovery

Causal inference concerns itself with identifying causal relationships from data [Pearl, 2000]. The central object of study is a Structural Causal Model (SCM), which characterizes the data generating distribution as a set of functions on observed and hidden variables.

Definition 8.2.1 ([Pearl, 2000])**.** *A structural causal model is a tuple* $(\mathbf{U}, \mathbf{V}, \mathcal{F}, P)$*, where* $\mathbf{U}$ *is a set of exogenous variables (e.g. the unobserved source of stochasticity in the environment) drawn from the distribution* P*,* $\mathbf{V}$ *is a set of endogenous variables (e.g. the observed state* s*, the reward* r*, and the action* a *in RL), and* $\mathcal{F}$ *is the set of functions* $f_V : \mathrm{PA}_V \times \mathbf{U} \to \mathbf{V}$ *with* $\mathrm{PA}_V \subset \mathbf{V}$*, which determine the value of endogenous variable* V *for each* $V \in \mathbf{V}$*.*

An SCM has a representation as a directed acyclic graph, called a causal Bayesian network or simply a causal graph, whose nodes are the variables in $\mathbf{U} \cup \mathbf{V}$ and

whose edges are given by $\{(p, v) \mid p \in \mathrm{PA}_V, V \in \mathbf{V}\}$. An edge from variable V_1 to V_2 indicates that V_1 is a causal parent of V_2. This mapping is not one-to-one. In general, many SCMs may induce identical causal graphs. Causal models enable reasoning about how changes to the data-generating process, called *interventions*, affect the resulting distribution over variables.

Definition 8.2.2 (Do-Intervention)**.** *A do-intervention on a variable V, denoted $do(V = v)$, in a causal model $S = (\mathbf{U}, \mathbf{V}, \mathcal{F}, P)$ is an operation that induces a new SCM $S' = (\mathbf{U}, \mathbf{V}, \mathcal{F}', P)$, where $\mathcal{F}' = \{f_W \in \mathcal{F} \mid W \neq V\} \cup \{f_{V=v}\}$ and $f_{V=v}(\mathbf{p}, \mathbf{u}) = v \; \forall \mathbf{p} \in \mathrm{PA}_V, \mathbf{u} \in U$.*

The randomized procedure used to assign patients to the treatment and control groups in medical trials is an example of an intervention. Predicting the effect of a do-intervention requires identifying the direction of the edges in the causal graph, a process known as *causal discovery*. Many recent works have explored how to incorporate causal structure into deep RL by learning disentangled representations of the observation space [Bengio et al., 2017, Suter et al., 2019].

8.2.2 Invariant prediction

A fundamental property of causal relationships is their invariance under interventions. *Invariant prediction* seeks to identify causal structure from data by evaluating whether the relationships between variables in a prediction problem are invariant across different environments, where each environment corresponds to a different intervention on the data generating process. Relationships which are invariant over these interventions are assumed to be causal, and by applying a straightforward hypothesis-testing approach [Peters et al., 2016], one can obtain high-probability guarantees on identifying the causal parents of a target variable of interest.

The approach of Peters et al. [2016] is limited to problems in which the inputs consist of sets of variables whose relationship can be described by a causal graph. A more adaptable framework based on the same principle of invariance has emerged recently: invariant risk minimization (IRM) [Arjovsky et al., 2019].

This setting assumes that the learner has access to data which is partitioned into different environments, and that these environments share the same causal structure. Differences in the distribution of data in each environment are attributed to non-causal correlations, which the learner should avoid using for prediction. Unlike invariant causal prediction methods, there is no explicit assumption on the structure of the inputs as corresponding to distinct variables in the causal model. Instead, the learner seeks to identify a non-linear feature map which induces a linear classifier whose errors are invariant over the training environments. The mathematical formalism of this objective yields the minimization problem

$$\min_{\phi} \sum_{e \in \mathcal{E}} \ell(w^\top \phi(X_e), Y_e) \text{ s.t. } \text{argmin}_v \ell(v^\top \phi(X_e), Y_e) = w, \ \forall e \in \mathcal{E} \ . \tag{8.1}$$

The invariant risk minimization framework provides a compelling intuition, but the IRM objective proposed by Arjovsky et al. [2019] requires extensive hyperparameter tuning to obtain competitive results, and in many cases fails to capture the invariances motivating its proposal [Kamath et al., 2021]. In spite of these limitations, several works have extended the ideas behind IRM to a variety of settings [Ahuja et al., 2020, Alesiani et al., 2021], and a notion of invariance across environments has recently been deployed by Raileanu and Fergus [2021] to improve generalization to novel environments on the ProcGen suite.

8.3 Problem setting

We consider a family of environments $\mathcal{M}_{\mathcal{E}} = \{(\mathcal{X}_e, \mathcal{A}, \mathcal{R}_e, \mathcal{P}_e, \gamma) \mid \ e \in \mathcal{E}\}$, where $\mathcal{E}$ is some index set. For simplicity of notation, we drop the subscript e when referring to the union over all environments $\mathcal{E}$, e.g. $\mathcal{X} = \cup_{e \in \mathcal{E}} \mathcal{X}_e$. Our goal is to use a subset $\mathcal{E}_{\text{train}} \subset \mathcal{E}$ of these environments to learn a policy π which generalizes to every environment. Concretely, we seek a parameterized policy π which maximizes the quantity

$$\mathbb{E}_{\mathcal{E}_{\text{test}}, \pi}[\sum_{t=0}^{\infty} \gamma^t R(x_t, a_t)] \ . \tag{8.2}$$

We denote the number of training environments by $N = |\mathcal{E}_{\text{train}}|$. We assume that the environments share some structure, and consider different degrees to which this structure may be shared in this section. As we have alluded to previously, it is this presumed structure which will enable significant sample efficiency gains over more naive approaches.

8.3.1 Block MDPs

Block MDPs [Du et al., 2019] are described by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{X}, p, q, R \rangle$ with a finite, unobservable state space $\mathcal{S}$, a finite action space $\mathcal{A}$, and a possibly infinite, but observable space $\mathcal{X}$. Here p denotes the latent transition distribution $p(s'|s, a)$ for $s, s' \in \mathcal{S}, a \in \mathcal{A}$, q is the (possibly stochastic) emission function that generates observations from the latent state $q(x|s)$ for $x \in \mathcal{X}, s \in \mathcal{S}$, and R denotes the reward function.

Assumption 2 (Block structure [Du et al., 2019])**.** *Each observation x uniquely determines its generating state s. That is, the observation space $\mathcal{X}$ can be partitioned into disjoint blocks $\mathcal{X}_s$, each containing the support of the conditional distribution $q(\cdot|s)$.*

This assumption gives us the Markov property in $\mathcal{X}$, and is crucial for our empirical and theoretical results. Zhang et al. [2020b] discuss the partially observable setting.

The definition of a block MDP given by Du et al. [2019] defines a single environment; our primary interest in this chapter, however is in a multi-environment setting. We will therefore concern ourselves with *block MDP families*, which characterize a set of environments with shared dynamics. We define a block MDP family as a collection of environments $\mathcal{M}_e$, where each environment e corresponds to an emission function q_e. Each environment $\mathcal{M}_e$ thus has the form $\langle \mathcal{S}, \mathcal{A}, \mathcal{X}, p, q_e, R \rangle$, where all terms except the emission function q_e are shared between environments. We will move the potential randomness from q_e into an auxiliary variable $\eta \in \Omega$, where Ω is some probability space, and write $q_e(s, \eta)$. Crucially, we enforce Assumption 2 on

the union of the emission functions q_e. This entails that whenever $\text{range}(q_e(s, \cdot)) \cap \text{range}(q_{e'}(s', \cdot)) \neq \emptyset$, then $s = s'$. The decomposition of the state into a latent state s and noise term η, and the corresponding causal structure, can be seen in Figure 8.1.

Without additional assumptions on shared structure between the emission functions, generalization to novel environments can be an impossible task. We will therefore focus on settings where the q_e overlap in structured ways – for example, where q_e outputs the concatenation of the noise and state variables, $q_e(s, \eta) = s \oplus f(\eta)$ – such that it is possible to learn a feature map from observations to latent states that will generalize to new emission functions from this class.

Our ultimate goal is to find a policy π which maximizes cumulative reward over any novel test environment. This chapter will exclusively consider settings where the policy π is a function of some state abstraction (also referred to as a feature map) $\phi(\mathcal{X})$. Under the block MDP assumption, if ϕ maps observations $\mathbf{x}$ to the state s that emitted them, then any policy which is optimal on the training environments will trivially be optimal on the test environments. As a result, most of our discussion in the coming sections will focus exclusively on the invariance of a learned state abstraction, with the ensuing optimality of the learned policy left implicit. Section 8.3.2 will characterize environments where learning such an invariant state abstraction is tractable.

8.3.2 Relaxations

Figure 8.1 illustrates the graphical model representing the transition dynamics of a Markov Decision Process under varying assumptions. This graphical model can be interpreted as a stationary causal graph, with arrows indicating causal dependence, as seen on the right hand side of the figure. Under this interpretation, different assumptions on the structure of the graph correspond to different classes of Markov Decision Process. We will be particularly interested in settings where different environments correspond to different correlation structure in the noise variables, as outlined below.

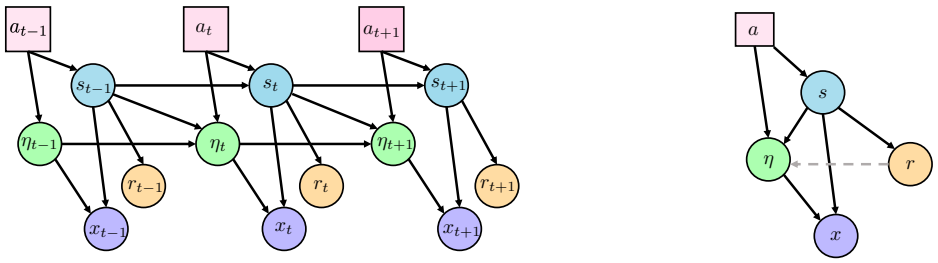


Figure 8.1: Graphical model of a block MDP with emissions x from a state which can be decomposed into a causal component s and a spurious component η. The left hand side illustrates the relationship between the variables at each time step, while the right hand side is a shorthand characterizing the same sequential distribution.

Correlated noise variables. The standard formulation of the block MDP assumes that the noise variable η is sampled independently at random at every time step, which prevents multi-timestep correlations. We therefore also consider a more realistic *relaxed block MDP*, where spurious variables may have different transition dynamics across the different environments so long as these dynamics are independent of the return. We introduce the notation q_S to refer to the projection of the inverse mapping q_e onto the invariant component of the latent state s,

$$q_S(x) = s \text{ where } \exists \eta, e \text{ s.t. } : q_e(s, \eta) = x \tag{8.3}$$

and note that this will be invariant over environments. This formulation is equivalent to augmenting each MDP $\mathcal{M}_e \in \mathcal{M}$ with a noise variable η_e, such that for any observations $x = q_e(s, \eta_e)$, any observation $x' = q_e(s', \eta'_e)$, and any action a, we have

$$p(x'|x, a) = p(s'|s, a) \int_{\eta'_e : q(q_S(x'), \eta'_e) = x'} p_e(\eta'_e|s, \eta_e) d\eta'_e \,.$$

This dependency can be observed in the arrows between η_{t-1} and η_t in Figure 8.1. We note that $p(s'|s, a)$ remains independent of η (i.e. there is no edge between η_t and s_{t+1}), while η_{t+1} is permitted to depend on s_t and η_t (i.e. we permit edges between these nodes in the causal graph). The assumption that for each observation x there is a unique generating latent state s is crucial to ensure that the return will be

independent of the variable η, and its absence results in a partially observable MDP. In principle, we might go further to relax the independence of η_t on the previous timestep to also causally depend on the reward (the grey dashed line in Figure 8.1); while the techniques shown in the subsequent sections can be applied in this setting, this chapter will focus on dependencies only between components of the state.

8.3.3 Assumptions on causal structure

Importantly, under Assumption 2 or under the correlated noise variables assumption, the spurious variables η_t are disjoint from the set of causal ancestors of the reward (i.e. the set of variables X from which there is a directed path in $\mathcal{G}$ from X to r_t for some t). The suggestive notation of s_t and η_t to describe the environment-specific and environment-invariant components of the state is deliberate: we will show later that identifying the abstraction $(s_t, \eta_t) \to s_t$ is equivalent to a causal structure identification problem, and that this abstraction guarantees optimal generalization to new environments.

In order to obtain these results, we must enforce the Markov property on the MDP's causal graph. This goes beyond the requirement that the variables in the environment state at time t can only affect the values of the state at time $(t+1)$, and can only affect the reward at time t, allowing us to consider the state and action at time t as the only candidate for causal parents of the state at time $(t+1)$ and of the reward at time t. We refer the reader to Figure 8.2 to demonstrate how causal graphical models can be translated to this setting.

Assumption 3 (Independent Causal Mechanisms)**.** *Let x^1 and x^2 be components of the observation $\mathbf{x}$. Let X_t^i denote the value of x^i at time t for $i = 1, 2$. Then when no intervention is performed on the environment, we have the following independence,*

$$X_{t+1}^1 \perp X_{t+1}^2 \mid \mathbf{x}_t \,. \tag{8.4}$$

The assumption of independent causal mechanisms is standard in the causal inference literature [Peters et al., 2017] but is stronger than the assumption of Markovian transition dynamics in the MDP. For example, an MDP whose state

space consists of two variables x_1 and x_2 which are both set at each timestep by the same coin flip will have Markov transition dynamics, but its causal graph will not satisfy the Markov condition. Intuitively, Assumption 3 requires that the variables in the observation space reflect the environment's underlying data generating process. This assumption is crucial for our results in linear function approximation, though our results on the rich observation setting allow us to relax independence of causal mechanisms in the observation space.

While many of the most challenging technical problems in causal inference stem from more complex causal structures which do not satisfy the Markov property, these structures often arise in the first place because the data collector is not able to see the step-by-step evolution of the system. Put simply: an RL agent can observe that rain always occurs before a person opens their umbrella, while a supervised predictor will have access to only an instantaneous snapshot of the world with binary values of rain and umbrella. As a result, RL agents have access to much richer temporal information about the data-generating distribution, and this removes many, but not all, of the ambiguities that causal inference methods seek to resolve.

We have thus far required that the dynamics of the latent states in the environment be independent of the spurious variables; however, the spurious variables may still correlate with the return to an extent that a function approximator may use them to predict the value function or to select an action. For example, the spurious variables may be a copy of the latent state at the previous time step. In order to identify the relevant and irrelevant components of the state, the agent must receive suitable information from the training environments. In this chapter we will assume that the training environments are generated by interventions on the data-generating process shown on the right hand side of Figure 8.1.

Assumption 4 (Environment Interventions)**.** *Let* $\mathcal{X} = X_1 \times \cdots \times X_n$*, and* $\mathcal{S} = X_{i_1} \times \ldots X_{i_k}$*. Each environment* $e \in \mathcal{E}$ *corresponds to a do-intervention or soft intervention [Eberhardt and Scheines, 2007] on a single variable* x_i *in the observation space.*

We note that in order for the training environments to constitute a block MDP family, these interventions may only be applied to the spurious variables in the state if they are applied over multiple timesteps. An intervention on the causal variables – for example, fixing the angle of a joint in a robotic simulator – will change the dynamics on the shared latent states $\mathcal{S}$. While such interventions may provide valuable information about the structure of the world, they will also change the value function that the agent is trying to predict. Interventions which only influence the initial state distribution do not encounter this issue, but face the limitation of providing only a single transition where the target variable takes the desired value. As a result, we assume that the test environments will only contain interventions on spurious variables, but allow for potential interventions on causal ancestors of the reward in the training environments.

8.4 State abstractions and causal feature sets

We begin our analysis with a simplified setting: we assume that the observation space $\mathcal{X}$ of the block MDP is a direct sum of variables $X_1, \ldots, X_n$, and that each of these variables can be represented as a node in a causal graph corresponding to the dynamics of the MDP which satisfies Assumption 3. Let $I \subset [n]$ be an index set of the variables which correspond to the latent state s in the block MDP. Then the state can be decomposed into $\mathbf{x} = (\mathbf{x}_I) \oplus (\mathbf{x}_{I^C}) = s \oplus \eta$, where $s = \mathbf{x}_I$ denotes the latent state and $\eta = \mathbf{x}_{I^C}$ denotes the spurious variables. This setting admits a natural formulation of invariant causal prediction in order to identify the state abstraction $\phi(\mathbf{x}) = \mathbf{x}_I$. It is also straightforward to alternate between the causal graph formulation of the MDP and the block MDP formalism, noting that the emission function q_e for an environment e will simply be of the form $q_e(s, \eta) = s \oplus \eta = \mathbf{x}_I \oplus \mathbf{x}_{I^C}$. We can further show that the state abstraction $\mathbf{x} \mapsto \mathbf{x}_I$ is a model-irrelevance state abstraction (c.f. Definition 2.3.1) for all $\mathcal{M}_e \in \mathcal{M}$.

Theorem 8.4.1 (Existence of model-irrelevance state abstractions)**.** *Let $\mathcal{M}_\mathcal{E}$ denote a block MDP family with joint observation space $\mathcal{X}_\mathcal{E} = \cup_{e \in \mathcal{E}} X_e$, where the observation*

space X_e is the image of some emission function $q_e(s, \eta)$ s.t. $s \in \mathcal{S}, \eta \in \Omega$ as before. Let $f_e = q_S \mid_{\mathcal{X}_e}$ as defined in (8.3). Then $\phi = \cup_{e \in \mathcal{E}} f_e$ is a model-irrelevance state abstraction for each $\mathcal{M}_e \in \mathcal{M}_{\mathcal{E}}$.

Proof. First, note that $\cup_{e \in \mathcal{E}} f_e$ is well-defined by the block MDP assumption. By assumption we have that the reward function satisfies $R(x) = R(q_S(x)) = R(\phi(x))$ for all observations x and environments e. We further have that for any x_1, x_2 s.t. $\phi(x_1) = \phi(x_2) = s$, and for any s':

$$\int_{x' \in \phi^{-1}(s')} P(x'|x_1, a) = P(s'|s, a) \int_{x' \in \phi^{-1}(s')} \int_{\eta' : q_e(s', \eta) = x'} P(\eta'|s, \eta_1) \tag{8.5}$$

$$= P(s'|s, a) \int_{x' \in \phi^{-1}(s')} \int_{\eta' : q_e(s', \eta) = x'} P(\eta'|s, \eta_2) \tag{8.6}$$

$$= \int_{x' \in \phi^{-1}(s')} P(x'|x_2, a) \tag{8.7}$$

Noting that $\phi(q_e(s, \eta')) = s$ and $\phi^{-1}(s) = \{q_e(s, \eta) \mid \eta \in \Omega\}$, we obtain the desired result. □

We now consider whether, under Assumptions 2, 3, and 4, a model-irrelevance state abstraction can be obtained by causal inference methods. Intuitively, one would then expect that the causal variables (in this case, the causal *ancestors* of the return, rather than the parents of the prediction target as in the case of regression problems) should have nice properties as a state abstraction: in particular, they should enable generalization to new environments. In what follows, we will generalize our definition of a model-irrelevance state abstractions to refer to a block MDP family for which the dynamics of the induced abstract MDP on each environment in the family are equivalent.

Definition 8.4.1. *A model-irrelevance state abstraction over a block MDP family $\mathcal{M}_{\mathcal{E}}$ is one for which the following two conditions hold. First, ϕ is consistent with the reward function R, i.e.*

$$R(\phi(\mathbf{x})) = R(\phi(\mathbf{y})) \quad \forall \mathbf{x}, \mathbf{y} \in \cup_{e \in \mathcal{E}} \mathcal{X}_e \, . \tag{8.8}$$

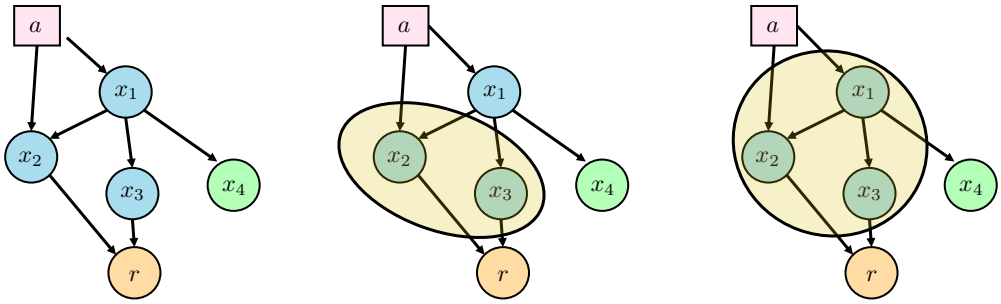


Figure 8.2: Illustrations of a causal state abstraction where the observation is a concatenation of variables from a causal graph. The variables $x_{1:3}$ present the causal ancestors of the reward, and as per Theorem 8.4.2 can be interpreted as the latent state s from Figure 8.1. The state x_4 thus corresponds to the spurious variable η.

Further, for all $e_1, e_2 \in \mathcal{E}, \mathbf{x}_1 \in e_1, \mathbf{x}_2 \in e_2, \mathbf{y} \in \mathcal{X}_{e_1} \cup \mathcal{X}_{e_2}$ *such that* $\phi(\mathbf{x}_1) = \phi(\mathbf{x}_2)$ *we have:*

$$\sum_{\mathbf{x}' \in \phi^{-1}(\mathbf{y})} P_{e_1}(\mathbf{x}'|\mathbf{x}_1) = \sum_{\mathbf{x}' \in \phi^{-1}(\mathbf{y})} P_{e_2}(\mathbf{x}'|\mathbf{x}_2) \tag{8.9}$$

The following result highlights the connection between the causal ancestor set and model irrelevance: a state abstraction that selects the set of causal variables from the observation space of a block MDP will be a model-irrelevance abstraction for every environment $e \in \mathcal{E}$ in this stronger sense.

Theorem 8.4.2. *Consider a family of MDPs* $\mathcal{M}_\mathcal{E} = \{(\mathcal{X}, A, \mathcal{R}, P_e, \gamma) | e \in \mathcal{E}\}$*, with* $\mathcal{X} = X_1 \times \cdots \times X_k$ *where each* $X_i, i \in 1, \ldots, k$ *is some collection of sets, for which there exists a SCM* $\mathcal{C} = ((x_t^1, \ldots, x_t^k, A_t, R_t)_{t=1}^\infty, \{\eta_1, \ldots, \eta_k\}, \mathcal{F}, P)$ *such that each* $\mathcal{M}_e \in \mathcal{M}_\mathcal{E}$ *corresponds to an intervention on a single variable* X_i *in* $\mathcal{C}$*, where* X_i *is not a causal ancestor of the reward. Let* $M_\mathcal{E}$ *satisfy Assumptions 2-4. Let* $S = \boldsymbol{AN}(R)$ *denote the ancestors of the reward* R *in* $\mathcal{C}$*. Then the state abstraction* $\phi_S(x) = [x]_S = (x_i \mid i \in S)$ *is a model-irrelevance abstraction over the block MDP family* $\{\mathcal{M}_e \in \mathcal{E}\}$*.*

Proof. This result follows as a special case of Theorem 8.4.1 after showing that $\mathcal{M}_\mathcal{E}$ is a block MDP family. To do so, we construct each component of such a block MDP family as follows: we set $\mathcal{S} = (X_i)_{i \in S}$, $\mathcal{A} = A$, $R = \mathcal{R}$ and $\mathcal{X} = \mathcal{X}$. We

define the transition operator $p = P_S$, where P_S denotes the marginal transition probability $P(X_S^{t+1} \mid X_S^t)$ which is well-defined as all causal parents of nodes in X_S are contained in X_S. The emission function q is defined on the joint space X_S and X_{S^C} via concatenation and re-ordering such that $q(s, \eta_e) \to x$ whenever $s = [x]_S$ and $\eta_e = [x]_{S^C}$. Crucially, the criterion that each environment correspond to an intervention requires that η_e evolve over time according to the dynamics induced by P_e for each environment e. However, because S is closed under taking causal parents and because Assumptions 1 and 2 apply, this correlation of the noise variable η_e^t over time will not interfere with the dynamics on $\mathcal{S}$. We thus obtain a (relaxed) block MDP family which precisely corresponds to $\mathcal{M}_\mathcal{E}$. This then implies that the map $\phi(x) = [x]_S$ is a model-irrelevance state abstraction. □

An important detail in the previous result is that the model-irrelevance state abstraction incorporates not just the parents of the reward, but also its ancestors. This is because in RL, we seek to model *return* rather than solely rewards, and a variable which is a causal parent of the return may exert influence over several timesteps. We provide an illustration of such a state abstraction in the rightmost column of Figure 8.2. As a concrete example from the CartPole environment, only the position x and pole angle θ are necessary to predict the reward. However, predicting the return requires knowledge of $\dot{\theta}$ and $\dot{x}$, their respective velocities.

Key insight.
To obtain a state abstraction which will be a model-irrelevance abstraction on all environments e in a block MDP, it suffices to identify the variables which are causal ancestors of the reward in the MDP's causal model.

Strictly speaking, other subsets of variables which include the set of causal ancestors S will be a model-irrelevance abstraction for each individual environment. What makes the set of causal ancestors S special is that it applies to all environments in the block MDP. This property is crucial for generalization: under $\phi : \mathbf{x} \to [\mathbf{x}]_S$, we have that the union over environment transition operators acting on the latent space $\phi(\mathcal{X})$ is well-defined. In other words, ϕ maps each environment to an abstract MDP whose dynamics and state space are identical across the block MDP family.

Under this definition a model-irrelevance state abstraction must remove all spurious variables from the input. Doing so is a non-trivial task, and requires observing the effects of their variation. It is not in general possible to identify spurious variables from only a single environment, particularly when spurious variables are highly correlated with their causal counterparts. The following proposition highlights the importance of Assumption 4 to the identifiability of the causal feature set. This result will require either that we have at least one environment corresponding to an intervention on each variable in the causal graph, or that we have a particular class of intervention applied to all variables simultaneously [Peters et al., 2016].

Proposition 8.4.3 (Identifiability of causal state abstractions)**.** *Consider a block MDP family whose corresponding SCM and set of training environments $\mathcal{E}$ satisfy the conditions of Theorem 2 of Peters et al. [2016]. Then the causal feature set ϕ_S is identifiable and corresponds to a model-irrelevance state abstraction for all test environments consisting of interventions on non-ancestors of the reward. Conversely, if these conditions are not satisfied then there may exist a state abstraction $\phi : \cup_{e\in\mathcal{E}}\mathcal{X}_e \to \bar{\mathcal{X}}$ such that ϕ is a model-irrelevance abstraction over $\mathcal{E}_{train}$, but not over $\mathcal{E}$ globally.*

Proof. The proof of the first statement follows immediately from the iterative application of the identifiability result of Peters et al. [2016] to each variable in the causal variables set. These conditions require that we also show the agent interventions on the *causal* variables; the resulting training environments will not necessarily satisfy the block MDP assumption, but may still be used for identification. Once a causal variable set has been identified, it is straightforward to use this variable set to train a value function on environments in the block MDP family. Alternatively, it is possible to show that for environments satisfying Assumption 3, so long as there is a do-intervention on each variable in $\mathbf{AN}(R)^C$ in the training set $\mathcal{E}$, we preserve identifiability of the causal variable set. Constructing this set requires a

slight modification of the aggregation procedure of Peters et al. [2016] to return the largest set S whose induced predictor is invariant.

For the converse, we consider a simple counterexample in which one non-ancestor of the return $\mathbf{x}_m$ is constant in every training environment with value v_m, where $\mathbf{x} = (\mathbf{x}_i)_{i=1}^d \in \mathcal{X}$. Then letting $S = \mathbf{AN}(R)$, we observe that $\phi = \mathbf{x} \mapsto \mathbf{x}_{S \cup \{m\}}$ is also a model-irrelevance state abstraction for each training environment $\mathcal{M}_e$. However, it is straightforward to show that if x_m is perturbed by a Bernoulli noise variable $Z \sim \text{Ber}(0.5)$ in a new test environment, then these transition dynamics will not satisfy the conditions for ϕ to be a model-irrelevance state abstraction in the new environment. Indeed, we now have that the state abstraction ϕ violates the second condition of Definition 8.4.1. In particular, let $\mathbf{x}$ be a state occurring in both a training environment e_{train} and the test environment e_{test}. Let $\mathbf{y} \in \mathcal{X}_{e_{\text{train}}}$ be such that $\mathbf{y}_m = v_m$, $P_{e_{\text{train}}}(\mathbf{y} \mid \mathbf{x}) > 0$, and $P_{e_{\text{test}}}(\mathbf{y} \mid \mathbf{x}) > 0$. Then

$$\sum_{\mathbf{x}' : \phi(\mathbf{x}') = \phi(\mathbf{y})} P_{e_{\text{train}}}(\mathbf{x}' | \mathbf{x}) = \sum_{\mathbf{x}' : \phi(\mathbf{x}') = \phi(\mathbf{y})} 2 P_{e_{\text{test}}}(\mathbf{x}' | \mathbf{x}) \; . \tag{8.10}$$

Because $P_{e_{\text{test}}}$ spreads probability mass equally over states where $\mathbf{x}_m = v_m$ and $\mathbf{x}_m = v_m + 1$. This violates our requirement that the union over transition operators $P_{e_{\text{train}}}$ and $P_{e_{\text{test}}}$ yield a well-defined function. □

8.4.1 Variable selection for linear predictors

We have shown that identifying a model-irrelevance state abstraction that will generalize to any environment consistent with the block MDP is possible by identifying the causal feature set. The following *Linear MISA* algorithm (Algorithm 8.4.1) provides a concrete approach to do this. We require the presence of a replay buffer $\mathcal{D}$, in which transitions are stored and tagged with the environment from which they came. The algorithm then applies ICP to find all causal ancestors of the reward iteratively. This approach has the benefit of inheriting many desirable properties from ICP – under suitable identifiability conditions, it will return the exact causal variable set to a specified degree of confidence.

Algorithm 8.4.1: Linear MISA: Model-irrelevance State Abstractions

Input: α, a confidence parameter; $\mathcal{D}$, a replay buffer with observations $\mathcal{X}$.

Output: $S \subset \{1, \ldots, k\}$, the causal state variables.

$S \leftarrow \emptyset$;
stack $\leftarrow$ r ;
while *stack is not empty* **do**
  v = stack.pop() ;
  if $v \notin S$ **then**
    $S' \leftarrow$ `ICP`(v, $\mathcal{D}$, $\frac{\alpha}{\dim(\mathcal{X})}$) ;
    $S \leftarrow S \cup S'$;
    stack.push(S')
  end
end

It also inherits inconvenient properties: the ICP algorithm is exponential in the number of variables, and so this method is not efficient for high-dimensional observation spaces. We are further restricted to considering linear relationships of the observation to the reward and next state. Additionally, because we take the union over iterative applications of ICP, the confidence parameter α used in each call must be adjusted accordingly. Given n observation variables, we use a conservative value of $\frac{\alpha}{n}$ to get an overall confidence of α from the procedure.

Evaluation

We illustrate the benefits of explicitly removing spurious features from the state input in Figure 8.3. We consider a simple family of MDPs with state space $\mathcal{X} = \{(x_1, x_2, x_3)\}$, with a transition dynamics structure such that $x_1^{t+1} = x_1^t + \epsilon_1^e$, $x_2^{t+1} = x_2^t + \epsilon_2^e$, and $x_3^{t+1} = x_2^t + \epsilon_3^e$. The reward $r(x)$ is set to be a linear function of x_1 and x_2. We train on 3 environments with soft interventions on each noise variable. We run the linear MISA algorithm on batch data from these 3 environments to get a state abstraction $\phi(x) = \{x_1, x_2\}$, then train 2 linear predictors on $\phi(x)$ and x. We evaluate the generalization error on novel environments that correspond to different hard interventions on the value of the x_3 variable. We observe that the predictor trained on $\phi(x)$ attains zero generalization error because it zeros out x_3 automatically. However, any nonzero weight on x_3 in the least-squares

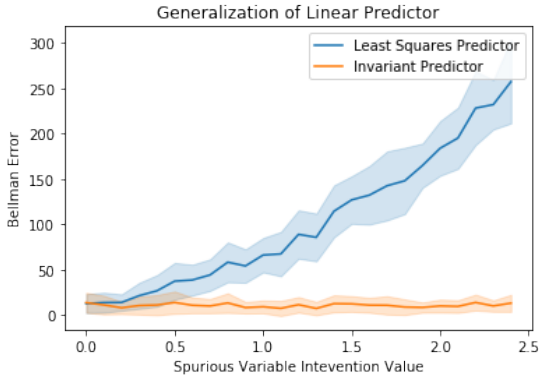


Figure 8.3: The presence of spurious uncorrelated variables in the state can still lead to poor generalization of linear function approximation methods. Figure shows Bellman error of linear function approximators in the presence of interventions. The linear model which is only given access to the invariant features obtained by Linear MISA is robust to these interventions, while the model trained on the entire input space exhibits increasing error as the intervention value grows.

predictor will lead to arbitrarily large generalization error, which is precisely what we observe in Figure 8.3.

8.5 State abstractions for rich observations

The variable selection problem has been an enlightening regime in which to ground our intuition about the relationship between invariant prediction and generalization. However, most RL problems of interest involve more complex observation spaces where simply identifying an input mask is not sufficient to guarantee generalization to new environments. This section leverages the intuition discussed previously to present a method motivated by invariant prediction which learns approximate model-irrelevance state abstractions in rich observation settings.

8.5.1 Block MDP generalization bounds

Our goal in this section is to *learn* representations that will generalize from the training environments to a novel test environment, as opposed to identifying input masks as we did in the previous section. However, normal PAC generalization bounds require a much larger number of environments than one could expect to obtain in the reinforcement learning setting. The appeal of an invariant representation is

that it may allow for theoretical guarantees on learning the right state abstraction with many fewer training environments, as discussed by Peters et al. [2016]. If the learned state abstraction is close to capturing the true latent dynamics of the block MDP family, then the model error, and by extension performance, in the test environment can be bounded by a function of the distance between the test environment's abstract state distribution and that of the training environments. In particular, if the state abstraction is invariant and always maps equivalent states in any pair of environments to the same latent representation, then the learned model error on the test environment will be equal to that on the training environments.

A more interesting case arises when the state abstraction ϕ is approximately invariant: such a state abstraction might map equivalent states from the training environment M and test environment M' to similar but non-identical feature vectors. For simplicity we will consider only deterministic environments, but similar reasoning can be applied to stochastic MDPs. In this case, a looser requirement on ϕ can still yield reasonable generalization error bounds for a learned transition dynamics model. Intuitively, the requirement we set is that the training environment dynamics function T_M be Lipschitz with respect to ϕ, and that the union of the training and test environment dynamics $T_M \cup T_{M'}$ be well-defined and Lipschitz. This is analogous to the requirement on the induced dynamics of a model-irrelevance state abstraction: if states from M and M' map to the same feature vector, then the successor states to each must also map to identical feature vectors. The Lipschitz requirement introduces a geometric notion of *similarity*: if states from M and M' map to nearby feature vectors, then their successors must also be relatively close in feature space. With this requirement, it is possible to bound the generalization error of a learned latent dynamics model trained on M in terms of the Lipschitz constant of $T_M \cup \mathcal{T}_{M'}$ and the Wasserstein distance between the state visitation distributions of the two environments in feature space. Intuitively, the Lipschitz constant quantifies the smoothness the environment dynamics with respect to ϕ, and the Wasserstein distance between the state visitation distributions of the two environments in feature space, which measures how well ϕ has been able to discard

irrelevant information. The following result assumes a fixed policy and a state-valued input to the transition function, but can be easily adapted to the control setting.

Theorem 8.5.1. *[Model error bound] Let $M = \langle \mathcal{X}, A, R, P, \gamma \rangle$ and $M' = \langle \mathcal{X}', A, R', P', \gamma \rangle$ be two environments from a block MDP family, and let $\phi : \mathcal{X} \to \mathbb{R}^d$ denote a model-irrelevance state abstraction that maps states from both MDPs to feature vectors. Set $T_M : \mathcal{X} \to \mathcal{X}$ as the (deterministic) transition function of M. Suppose that the union of the dynamics of M and M' are L-Lipschitz with respect to the embedding ϕ and that $T : \mathbb{R}^d \to \mathbb{R}^d$ is some approximate transition model satisfying $\max_s \mathbb{E}\|T(\phi(x)) - \phi(T_M(x))\| < \delta$, for some $\delta > 0$. Let $W_1(\pi_1, \pi_2)$ denote the 1-Wasserstein distance between distributions π_1, π_2 on $\mathbb{R}^d$. Then*

$$\mathbb{E}_{x \sim M'}[\|T(\phi(x)) - \phi(T_{M'}(x))\|] \leq \delta + 2LW_1(\pi_{\phi(M)}, \pi_{\phi(M')}). \tag{8.11}$$

Proof. The proof follows from a straightforward decomposition of the error into three components: one which depends on the similarity between each input $x \in X_{M'}$ and the nearest neighbour y in X_M, the model error on y, and the similarity between the representations of the successor states $T_M(x)$ and $T_M(y)$:

$$\begin{aligned}
&\mathbb{E}_{x \sim M'}[\|T(\phi(x)) - \phi(T_{M'}(x))\|] \\
&= \mathbb{E}_{x \sim M'}\left[\min_{y \in X_M} \|T(\phi(x)) - T(\phi(y)) + T(\phi(y)) - \phi(T_M(y)) + \phi(T_M(y)) - \phi(T_{M'}(x))\| \right] \\
&\leq \mathbb{E}_{x \sim M'}\left[\min_{y \in X_M} \|T(\phi(x)) - T(\phi(y))\| + \|T(\phi(y)) - \phi(T_M(y))\| + \|\phi(T_M(y)) - \phi(T_{M'}(x))\| \right]
\end{aligned}$$

Letting γ be a coupling over the distributions of $\phi(\mathcal{X}')$ and $\phi(\mathcal{X})$ which minimizes the objective of the Wasserstein distance, i.e. $\mathbb{E}_{\gamma(\phi(x), \phi(y))}\|\phi(x) - \phi(y)\| = W_1(\pi, \pi')$

$$\begin{aligned}
&\leq \mathbb{E}_{\gamma(\phi(x), \phi(y))}\Big[\|T(\phi(x)) - T(\phi(y))\| + \delta + L\|x - y\| \Big] \\
&\leq \mathbb{E}_{\gamma(\phi(x), \phi(y))}\Big[L\|\phi(x) - \phi(y)\| + \delta + L\|\phi(x) - \phi(y)\| \Big] \\
&= \mathbb{E}_{\gamma(\phi(x), \phi(y))}\Big[L\|\phi(x) - \phi(y)\| + \delta + L\|\phi(x) - \phi(y)\| \Big] \\
&= 2LW_1(\pi, \pi') + \delta
\end{aligned}$$

□

Analogous results can be obtained for value error [Zhang et al., 2020a], which provide a bound on generalization performance that depends on the supremum of the dynamics and reward errors obtained by a learned model, along with the degree of invariance captured by the representation. The former corresponds to the empirical risk in a PAC bound, while the latter corresponds to the *generalization gap*. A state abstraction which perfectly captures the equivalence classes of the block MDP family will attain error on new environments equal to that of its error on the training environments and exhibit no generalization gap. Meanwhile, a state abstraction equal to the identity map can induce arbitrarily large errors on test environments.

8.5.2 Learning a model-irrelevance state abstraction

It is clearly desirable to learn an invariant representation across environments. How to do so is less clear. In this section, we propose one such approach based on a model-learning objective, but note that the principles guiding this method can easily be applied to other representation-learning methods. Our approach aims to learn a representation $\phi : \mathcal{X} \to \mathbb{R}^d$ which preserves the reward- and dynamics-relevant information of the state, while discarding environment-specific information in settings where such information is encoded non-linearly in rich observations. Ideally, a representation which discards environment-specific information in the training environments should also do so in new test environments, though guaranteeing this property is more challenging than in the variable selection setting.

We now present an objective to learn a dynamics preserving state abstraction $\mathcal{Z}$. This requires disentangling the state space into a minimal representation that causally relates to the reward $s_t := \phi(x_t)$ and all other features of the observation $\eta_t := \varphi(x_t)$.

We decompose our learning objective into two parts:

- **Invariance**: we train a **task classifier** on the shared latent representation $C : \mathcal{Z} \mapsto [0, 1]^N$ with cross-entropy loss and employ an adversarial loss [Tzeng et al., 2017] on ϕ to maximize the entropy of the classifier output to ensure task specific information is not passing through to $\mathcal{Z}$.

- **Dynamics preservation:** we train a reward model and a latent dynamics model on the state abstraction φ to encourage the preservation this information.

  The **reward model** is an MLP R trained to predict the sample reward r given input transition $(\phi(\mathbf{x}), a, \phi(\mathbf{x}'))$, using the objective

$$J_R(\phi, R) = \sum_i \mathbb{E}_{\pi_{b_i}}\left[(R(\phi(x_i), a, \phi(x_i')) - r_i')^2\right].$$

  The **transition model** is an MLP f_s acting on the latent space $\mathcal{Z}$. The transition model is trained via a reconstruction loss, using a learned decoder ϕ^{-1} which takes as input a task-dependent state embedding $\psi(\mathbf{x})$ along with the latent-space model output $f_s(a, \phi(\mathbf{x}))$, and outputs a predicted next observation.

$$J_D(\phi, \psi, f_s, f_\eta) = \sum_i \mathbb{E}_{\pi_{b_i}}\left[(\phi^{-1}(f_s(a, \phi(x_i)), f_\eta(a, \psi(x_i))) - x_i')^2\right],$$

This gives us a final objective

$$J_{\text{ALL}}(\phi, \psi, f_s, f_\eta, r) = J_D(\phi, \psi, f_s, f_\eta) + \alpha_R J_R(\phi, r) - \alpha_C H(C(\phi)), \tag{8.12}$$

where α_R and α_C are hyperparameters and H denotes entropy (Algorithm 8.5.1).

8.5.3 Results

We first evaluate MISA in a rich-observation model-learning task to evaluate whether the learned representation and abstract model do indeed generalize to complex dynamics structures where the factors of variation correspond to non-linear functions of the observation. We next look to imitation learning (Section 8.5.4), where we show that the state abstractions learned by MISA allow the agent to zero-shot generalize to observations generated by new camera angles. Finally, we explore end-to-end reinforcement learning in the low-dimensional observation setting with correlated noise (Section 8.5.5) and again show generalization capabilities where single task and multi-task methods fail.

Algorithm 8.5.1: Nonlinear Model-irrelevance State Abstraction (MISA) Learning

Input: $\mathcal{E}$, a set of environments; π_0, an initial policy; ϕ_0, f_0 an initial invariant feature map and transition model; $\{\psi_0^e \mid e \in \mathcal{E}\}$ and $f_{\eta,0}^e$, initial task-specific feature maps and transition models.

Output: ϕ, an invariant state encoder.

$\pi \leftarrow \pi_0$;
$\phi, f_s \leftarrow \phi_0, f_{s,0}$;
$\psi^e, f_\eta^e \leftarrow \psi_0^e, f_{\eta,0}^e$ for $e \in \mathcal{E}$;
$\mathcal{D}_e \leftarrow \emptyset$ for $e \in \mathcal{E}$;
while *forever* **do**
  for $e \in \mathcal{E}$ **do**
    $a \leftarrow \pi(x_e)$;
    $x'_e, r \leftarrow \texttt{step}(x_e, a)$;
    $\texttt{store}(x_e, a, r, x'_e)$;
  end
  for $e \in \mathcal{E}$ **do**
    Sample batch X_e from $\mathcal{D}_e$;
    $f_\eta^e, \psi^e \leftarrow \nabla_{f_\eta^e, \psi^e}[J_D(X_e)]$;
  end
  $f_s, \phi, r \leftarrow \sum_{X_e} \nabla_{f_s,\phi}[J_{\text{ALL}}(X_e)]$;
  $C \leftarrow \nabla_C \texttt{CE_loss}(C(\phi(\{x_e\}_{e\in\mathcal{E}}), \{e\}_{e\in\mathcal{E}})$;
end

Model learning: rich observation setting

We evaluate the gradient-based MISA method (Algorithm 8.5.1) in a setting with nonlinear dynamics and rich observations. We set the agent the task of learning a model which can generalize to new background colours on pixel-valued control tasks. We randomly initialize the background colour of two training environments from Deepmind Control [Tassa et al., 2018] from the range $[0, 255]^3$. We also randomly initialize another two backgrounds for evaluation. The orange line in Figure 8.4 shows performance on the evaluation environments in comparison to three baselines. In the first, we train on a single environment and test on another with our method, (`MISA - 1 env`). Without more than a single experiment to observe at training time, there is no way to identify whether or not the background can safely be ignored, and indeed we observe that the method appears to include information about the background in its representation. In the second baseline,

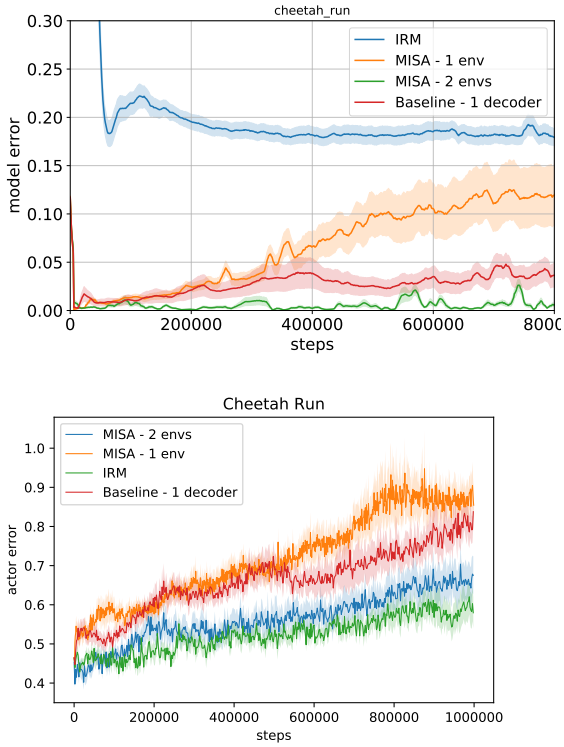


Figure 8.4: Evaluations over 10 seeds, with one standard error shaded. Left: Model error on evaluation environments on Cheetah Run from Deepmind Control. Right: Actor error on evaluation environments on Cheetah Run from Deepmind Control.

we combine data from the two environments and train a model over all data (`Baseline - 1 decoder`), but without explicitly encouraging invariance across the two environments. The third is another invariance-based method which uses a gradient penalty, IRM [Arjovsky et al., 2019]. In the second case the error is tempered by seeing variance in the two environments at training time, but it is not as effective as MISA with two environments. In the case of IRM, the loss starts much higher but very slowly decreases, and we find it is very brittle to tune in practice. Implementation details are deferred to Appendix F.1.1.

8.5.4 Imitation learning

Model-learning is often a useful auxiliary task in RL problems, but ultimately our the objective in RL is to learn an effective policy. We now evaluate how well a policy trained on top of the invariant representation found by MISA is able to generalize to new environments. We focus on imitation learning, as this setting involves fewer



Figure 8.5: The Cheetah Run environment from Deepmind Control with different camera angles. The first two images are from the training environments and the last image is from evaluation environment.

moving parts than an online learning setting, while preserving the policy-learning challenge of the RL problem. In this setup, we first train an expert policy using the proprioceptive state of Cheetah Run from the DeepMind Control suite [Tassa et al., 2018]. We then use this policy to collect a dataset for imitation learning in each of two training environments. When rendering these low dimensional images, we alter the camera angles in the different environments (Figure 8.5). We report the generalization performance as the test error when predicting actions in Figure 8.4. Model error on the evaluation environment increases significantly over the course of training in all baselines; while we see that test error does increase with MISA as well, the error growth is significantly slower compared to single task and multi-task baselines.

8.5.5 Reinforcement learning

Finally, we evaluate Algorithm 8.5.1 on the full RL loop. This presents a greater challenge than was observed in the offline learning settings, as the invariant prediction component of the model must now face the possibility that the learned policy will visit different state distributions in different environments, particularly early in training. We therefore go back to the proprioceptive state in the `cartpole_swingup` environment in DeepMind Control [Tassa et al., 2018] to show that we can in some cases still learn a model-irrelevance state abstraction while training a policy. We use the Soft Actor Critic (SAC) algorithm [Haarnoja et al., 2018] with an additional linear encoder, and add spurious correlated dimensions which are a multiplicative factor of the original state space. We also add an additional environment identifier to the observation. This factor varies across environments. We train on two

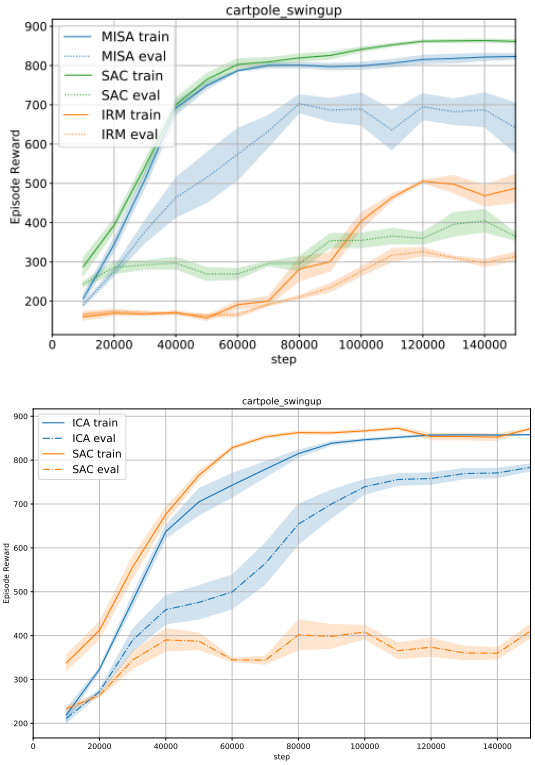


Figure 8.6: Left: generalization gap in SAC performance with 2 training environments on `cartpole_swingup` from DMC compared against two invariant prediction methods. Both experiments are evaluated with 10 seeds, standard error shaded. Right: generalization gap comparison between SAC and nonlinear MISA.

environments with multiplicative factors $1\times$ and $2\times$, and test on $3\times$. Like Arjovsky et al. [2019], we also incorporate noise on the causal state to make the task harder, specifically Gaussian noise $\mathcal{N}(0, 0.01)$ to the true state dimension. This incentivizes the agent to attend to the spuriously correlated dimension instead, which has no noise. In Figure 8.6 we see the generalization gap drastically improves with our method in comparison to training SAC with data over all environments in aggregate and with the IRM objective [Arjovsky et al., 2019] implemented on the critic loss. Intriguingly, we find that even without noise on the ground truth states, with only two environments, baseline SAC fails (Figure 8.6). Implementation details and more information about SAC can be found in Appendix F.1.2.

8.6 Conclusions

The findings presented in this chapter illustrate the importance of leveraging explicit assumptions on the causal structure of the environment for sample-efficient multi-environment generalization. We have shown that the application of invariant prediction methods in RL allows agents to learn policies that robustly generalize across environments with a shared causal structure after experiencing only a handful of training environments. We have further provided methods for both the low dimensional linear value function approximation setting and the deep RL setting which leverage invariant prediction to extract a causal representation of the state.

The block MDP family we defined in this chapter provides an explicit characterization of a rich class of MDPs in which generalization from a training task to a novel environment is tractable; however, promising future directions include extending these results to apply to an even broader class of problems. The proposed method in this chapter has not solved the generic problem of generalization to new environments in RL, but it has clearly demonstrated the benefits of applying the principle of invariant prediction to identify features that are useful for generalization in this expressive family of environments. Future work on this problem setting may find variations on invariant prediction which are more robust and which can handle the non-stationary rich-observation input distributions found in many deep RL problems. It may also extend these ideas to related problems including third-person imitation learning and sim-to-real transfer.

If the reader is to come away with one insight, it is this: just as representations which enable generalization between training inputs enable better generalization in the single task setting (as we saw in Chapters 4, 3, and 7), a representation which enables generalization between different training *environments* is likely to do the same in the multi-environment setting. Translating invariant prediction methods to RL is non-trivial; in large part this is due to the non-stationarities described in Chapters 5 and 6. However, our findings here present a striking proof-of-concept of the utility of these ideas in reinforcement learning.

9 Conclusion

It is easy to ensure that a deep neural network learns what we tell it to learn: doing so simply requires ensuring sufficient expressivity for the size of the training set at hand. It is an unsolved problem to ensure that it learns what we *want* it to learn, particularly in the context of deep reinforcement learning, where what we want a network to learn may evolve over the course of training. This thesis has argued that understanding how a network learns, by studying the properties of the learning trajectory taken over the course of training, is crucial to ensure that the learned function captures the underlying structure of the data it was trained on. We have applied this principle to a vast array of application areas, from Bayesian model selection to neural architecture search, from reinforcement learning to computer vision problems. Throughout these domains, a consistent trend has emerged: networks which are better able to use an update to their prediction on one data point to improve their accuracy on other data points train faster, generalize better, and have higher marginal likelihoods.

Discussion

Our primary quantity of interest has been interference, a notion of intra-dataset generalization. The intuition motivating this study is simple: a network which

can leverage information from a gradient update on one data point to improve its performance on other data points should generalize better than one which treats the data as being independent or, worse, faces competing interests wherein improving a prediction on one data point reduces performance on others. A model which experiences positive interference will converge more quickly and in principle generalize better than one which exhibits weak or conflicting interference. This intuition, inspired by a connection to the Bayesian marginal likelihood, motivated our study of training speed and invariance in Chapters 3 and 4. Many implications of this relationship have been leveraged in follow-up work to develop efficient performance estimators for neural architecture search and hyperparameter selection.

Whereas supervised learning involves fitting a single target function that remains fixed over the course of training, the targets that value-based RL objectives seek to fit are more mercurial, evolving alongside the network's predictions. This complicates the story relating training speed and generalization, and requires additional analytical and empirical tools. We developed a notion of subspace convergence to study the representations learned by an idealized model of deep RL agents in the theoretical results of Chapter 5. We went on to present a characterization of an agent's representation in terms of the types of functions it is biased towards fitting quickly in Chapter 6. These tools provided us with a number of fruitful insights into the learning dynamics of reinforcement learning agents, setting the stage for our study of generalization in Chapters 7 and 8.

Our study of generalization in deep RL leveraged these tools to understand the nuanced trade-offs between generalization and stability in value-based RL algorithms. This analysis hinged on a key insight outlined in Chapter 7: that the targets which deep RL agents seek to fit initially contain relatively little information about environment structure, and only accumulate this structure later in training. By asking networks to fit unstructured, discontinuous targets in the critical early phases of training, we bias them towards regions of the parameter space where interference between observations is weaker, reducing their potential for generalization. This analysis presented some obvious work-arounds in the form of

post-training distillation which we showed improve generalization and extrapolation, but the benefits of this approach were largely limited to generalization within a single environment. Chapter 8 concluded with discussion of generalization via invariant prediction, establishing connections to causality to address these limitations. We showed that by assuming a large degree of shared structure between environments, we can significantly improve upon the sample-efficiency of standard RL methods via an application of invariant prediction algorithms from causal inference.

The initial motivation of this thesis was to understand why neural networks generalize, and how this picture can become complicated in the setting of reinforcement learning. Looking back, it is clear that this was the wrong question: firstly, because as we have seen in this thesis neural networks often *fail* to generalize, particularly in the case of out-of-distribution data and in RL problems; secondly, because there is not in general a single straightforward explanation as to why a given neural network generalizes well or poorly. Instead, a variety of properties of a network architecture and training procedure come into play, and to understand the relationship between these properties and generalization it is crucial to understand the learning dynamics of the training procedure. This thesis has demonstrated the utility of this perspective in both supervised and reinforcement learning settings, and provided a number of theoretical and empirical tools for further analysis.

Further work

While there are many straightforward extensions of the results presented in the preceding chapters, a number of deeper questions also emerge concerning how to apply these ideas to the emerging training regimes for state-of-the-art models in visual and language domains. Chief among these is the recent success of pretraining models with billions or trillions of parameters via self-supervised learning on internet-scale corpuses of data. How should we evaluate interference in such cases? As opposed to the supervised learning regime of Chapters 3 and 4, state of the art training procedures involve showing the network a vast and diverse array of data and then fine-tuning its predictions on some downstream tasks. In these cases

the line between a model's inductive bias and its learned predictions becomes blurred, making the application of the tools connecting training speed and model selection presented here nontrivial. Interesting connections may arise between the analysis of Chapter 5 and the self-supervised learning objectives used in pretraining large models: temporal difference learning can be viewed as implicitly training the network to be self-predictive before any 'fine-tuning' reward signal is given. It also raises an interesting question of whether the ideas behind invariance and interference might be used to shape pretraining objectives which build even better inductive biases than what is used by autoregressive models.

A second cluster of intriguing ideas concern trade-offs between plasticity and generalization, and the precise relationship between loss of plasticity and declining generalization performance. Chapter 6 suggests that in many cases the correlation between the rank of a feature embedding and the plasticity of the network which parameterizes it is weak. The degree to which the network disentangles inputs can measure not just its ability to linearly distinguish observations, but also the degree to which its predictions have needed to change in order to fit its previous targets. Intriguingly, this class of changes to the network's representation might not only be bad for plasticity, but also for generalization. A network which has learned to spread apart the feature representations of its observations will tend to exhibit less generalization between inputs. This presents a conflict between the intuitions of prior literature in reinforcement learning, that higher-dimensional embeddings should correspond to networks with greater flexibility, and that from deep learning, where significant changes to a network's outputs tend to reduce its plasticity and harm generalization.

However, we must also consider potential trade-offs between plasticity, generalization, and accuracy on the *current* task. It may be the case that in order to solve complex prediction tasks, a network must necessarily reduce its ability to quickly fit unrelated targets. Analogously, it may be the case that excessive generalization too early in training could destabilize learning, making memorization a feature rather than a bug of the learning process. If such trade-offs exist, they will carry significant

implications for the application of RL to increasingly complex tasks, where training trajectories are long and require a persistent ability to adapt to new information and generalize from old experiences. It may be the case that new architectures and training procedures are needed in order to avoid the pitfalls identified in this thesis, which take into account the continual nature of reinforcement learning problems.

Finally, the role of generalization between data points can likely be leveraged more explicitly in the training process to encourage the network to develop the correct inductive biases for the data set it is trained on. To some degree, the implicit bias of gradient descent appears to already push large models towards solutions which generalize well after the training accuracy has saturated in many natural language tasks, a phenomenon commonly referred to as 'grokking'. By more directly encouraging networks to pick up on features which generalize between many training inputs, it may be possible to accelerate this phenomenon to obtain solutions which generalize well earlier in training.

Beyond these concrete research directions, the findings presented in this thesis raise a deeper philosophical question. We discussed in Section 2.2.2 the distinction between mathematical truths and scientific theories, and have seen examples of each throughout the thesis. Our formal guarantees have consistently been forced to trade off between generality of the problem setting (e.g. the exact gradient updates of Theorem 5.4.1, the linear group action in Theorem 3.3.2, and the Lipschitz continuity of the transition operator in Theorem 8.5.1), and precision of the resulting guarantee. In contrast, many chapters have gone on to distill the intuition behind each theoretical result into a principle which yields empirically testable predictions, such as the training speed estimator of Chapter 4 and the nonlinear invariant prediction methods of Chapter 8. Arguably, it is the empirical test of the intuition which has provided the greatest practical insight into the phenomena of interest to this thesis. But is this empirical approach capable of truly giving an explanation of generalization?

On its face, the answer would appear to be negative. Our empirical approach to generalization is almost tautological. A network will generalize well to test data, we

have shown, if it generalizes well between data points it encounters during learning. However, the ultimate reason why a network generalizes on a given dataset is because its inductive bias is a good fit for a data. Developing a detailed understanding of the training trajectory is crucial to be able to obtain a measure of the fit between a network's inductive bias and the data generating distribution. This detailed understanding of a network's training trajectory has also shown to bring useful insights into more nuanced notions of generalization and capacity in reinforcement learning, where a network must not only generalize between observations drawn from some distribution, but also to new observations and new prediction targets. The learning dynamics framework has thus already born fruit in both providing insight into generalization under the classical i.i.d. assumption on the training and evaluation data, and broadening this notion of generalization to apply to a richer class of prediction problems arising in reinforcement learning. Whether or not the precise empirical methods and theoretical tools presented in this thesis are leveraged directly in future work, it is clear that the study of learning dynamics promises to provide deep insights into some of the most fundamental questions about generalization in the years to come.

Appendices

A

The role of invariance in generalization

A.1 Proofs

Proposition 3.3.1. *Let $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ be a loss function that is convex in its first argument. Then for any $f : \mathcal{X} \to \mathcal{Y}$,*

$$\widehat{R}_\ell(f^\circ, \mathcal{D}^n) = \widehat{R}^\circ_\ell(f^\circ, \mathcal{D}^n) \leq \widehat{R}^\circ_\ell(f, \mathcal{D}^n) \,,$$

and therefore analogous inequalities hold for $\widehat{R}_\ell(Q^\circ, \mathcal{D}^n)$, $R_\ell(f)$, and $R_\ell(Q)$. Furthermore, if $\ell(f(\bullet), \bullet) \in L_2(P_\mathcal{D})$ (i.e., has finite second moment), then

$$\mathrm{Var}_{\mathcal{D}^n \sim P_\mathcal{D}^n}\left[\widehat{R}_\ell(f^\circ, \mathcal{D}^n)\right] \leq \mathrm{Var}_{\mathcal{D}^n \sim P_\mathcal{D}^n}\left[\widehat{R}^\circ_\ell(f, \mathcal{D}^n)\right] \,.$$

Proof of Proposition 3.3.1. Let $\mathcal{G}$ be a group with some probability measure λ, and F a class of functions $f : \mathcal{X} \to \mathbb{R}$. Let $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ be a loss function such that $\ell(f(\bullet), \bullet) \in L_2(P_\mathcal{D})$ for every $f \in F$. Then the augmented risk of any function $f \in F$ is

$$\widehat{R}^\circ_\ell(f, \mathcal{D}^n) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{E}_{G \sim \lambda}[\ell(f(GX_i), Y_i)] \,.$$

If ℓ is convex in the first argument, then by Jensen's inequality,

$$\mathbb{E}_{G \sim \lambda}[\ell(f(GX_i), Y_i)] \geq \ell(\mathbb{E}_{G \sim \lambda}[f(GX_i)], Y_i) \,, \quad i = 1, 2, \ldots, n \,. \tag{A.1}$$

On the other hand, the $\mathcal{G}$-symmetrization of $f(X)$ is $f^\circ(X) = \mathbb{E}_{G\sim\lambda}[f(GX)]$, with augmented risk

$$\begin{aligned}\widehat{R}^\circ_\ell(f^\circ, \mathcal{D}^n) &= \frac{1}{n}\sum_{i=1}^n \mathbb{E}_{G\sim\lambda}[\ell(\mathbb{E}_{G\sim\lambda}[f(GX_i)], Y_i)] \\ &= \frac{1}{n}\sum_{i=1}^n \ell(\mathbb{E}_{G\sim\lambda}[f(GX_i)], Y_i) = \widehat{R}_\ell(f^\circ, \mathcal{D}^n)\ .\end{aligned}$$

Combined with (A.1), the reduction in empirical augmented risk follows. The reduction in $\widehat{R}^\circ_\ell(Q, \mathcal{D}^n)$ follows trivially.

The variance-reduction is established by extending the argument in the proof of Chen et al. [2019]. Specifically, by the conditional Jensen's inequality,

$$\begin{aligned}\mathrm{Var}_{\mathcal{D}^n\sim P_{\mathcal{D}}^n}\left[\widehat{R}^\circ_\ell(f, \mathcal{D}^n)\right] &= \mathrm{Var}[\mathbb{E}[\widehat{R}^\circ_\ell(f, \mathcal{D}^n) \mid \Phi^n]] \geq \mathrm{Var}[\mathbb{E}[\widehat{R}_\ell(f^\circ, \mathcal{D}^n) \mid \Phi^n]] \\ &= \mathrm{Var}_{\mathcal{D}^n\sim P_{\mathcal{D}}^n}\left[\widehat{R}_\ell(f^\circ, \mathcal{D}^n)\right]\ .\end{aligned}$$

□

Proposition 3.3.2. *Let $P_{\mathcal{D}}$ and $\mathbf{w}^*$ be defined as above, and let $\ell(\mathbf{w}) = \mathbb{E}_{\mathbf{x},y\sim P_{\mathcal{D}}}[(\langle \mathbf{w}, \mathbf{x}\rangle - y)^2]$. Then the (global) minimizer $\mathbf{w}^*$ of ℓ satisfies $\rho^*_g \mathbf{w}^* = \mathbf{w}^*$ for λ-almost all $g \in \mathcal{G}$. In particular, the function $f(\mathbf{x}) = \langle \mathbf{w}^*, \mathbf{x}\rangle$ is $\mathcal{G}$-invariant.*

Proof. Let $w \in V^*$, and suppose that w is not invariant under the action of $\mathcal{G}$. Let $w^\circ = \mathbb{E}_{G\sim\lambda}[\rho^*_G w]$, which is $\mathcal{G}$-invariantby construction. Because $\mathcal{X}$ spans V and , $w - w^\circ \neq 0$ implies that $w^\top(\mathcal{X}) \neq w^{\circ\top}(\mathcal{X})$.

Consider the minimizer

$$\hat{\mathbf{w}} = \mathrm{argmin}_{\mathbf{w}\in V^*}\widehat{R}^\circ_\ell(f_{\mathbf{w}}, \mathcal{D}^n) = \mathrm{argmin}_{\mathbf{w}\in V^*}\frac{1}{n}\sum_{i=1}^n \mathbb{E}_{G\sim\lambda}[\ell(\mathbf{w}^\top \rho_G X_i, Y_i)]\ ,$$

which is unique because ℓ is strictly convex by assumption. Assume that $\hat{\mathbf{w}}$ is not $\mathcal{G}$-invariant. Applying Jensen's inequality, we have

$$\begin{aligned}\widehat{R}^\circ_\ell(f_{\hat{\mathbf{w}}}, \mathcal{D}^n) &= \frac{1}{n}\sum_{i=1}^n \mathbb{E}_{G\sim\lambda}[\ell(\hat{\mathbf{w}}^\top \rho_G X_i, Y_i)] \\ &> \frac{1}{n}\sum_{i=1}^n \ell(\mathbb{E}_{G\sim\lambda}[\hat{\mathbf{w}}^\top \rho_G X_i, Y_i)] \\ &= \frac{1}{n}\sum_{i=1}^n \ell(\mathbb{E}_{G\sim\lambda}[(\rho^*_{G^{-1}}\hat{\mathbf{w}})]^\top X_i, Y_i)] \\ &= \frac{1}{n}\sum_{i=1}^n \ell(\hat{\mathbf{w}}^\circ X_i, Y_i) = \widehat{R}^\circ_\ell(f_{\hat{\mathbf{w}}^\circ}, \mathcal{D}^n)\ ,\end{aligned}$$

which cannot be the case because $\hat{\mathbf{w}}$ minimizes $\widehat{R}_\ell^\circ$. Therefore, $\hat{\mathbf{w}}$ must be $\mathcal{G}$-invariant.

□

Lemma 3.4.2. *Suppose that $(E_i, \mathcal{E}_i)$, $i = 1, 2$, are two measurable spaces, the second of which is standard, μ and ν are two probability measures on $(E_1, \mathcal{E}_1)$, and $\psi : (E_1, \mathcal{E}_1) \to (E_2, \mathcal{E}_2)$ is a measurable map. Then*

$$\mathrm{KL}(\mu \circ \psi^{-1} \,||\, \nu \circ \psi^{-1}) \leq \mathrm{KL}(\mu \,||\, \nu) \,. \tag{3.8}$$

Furthermore, if $\mu \ll \nu$ with density m, then $\mu \circ \psi^{-1} \ll \nu \circ \psi^{-1}$ with density m_ψ, and the ψ-gap is

$$\begin{aligned} \Delta_\psi(\mu \,||\, \nu) :&= \mathrm{KL}(\mu \,||\, \nu) - \mathrm{KL}(\mu \circ \psi^{-1} \,||\, \nu \circ \psi^{-1}) \\ &= \int_{E_1} \mu(dx) \log \frac{m(x)}{(m_\psi \circ \psi)(x)} \,. \end{aligned} \tag{3.9}$$

The proof of Lemma 3.4.2 relies on the chain rule of relative entropy. Let two probability measures, $\tilde{\mu} \ll \tilde{\nu}$ defined on the product space $(E_1 \times E_2, \mathcal{E}_1 \otimes \mathcal{E}_2)$, have marginal measures $\tilde{\mu}_1 \ll \tilde{\nu}_1$ on $(E_1, \mathcal{E}_1)$ (respectively, $\tilde{\mu}_2 \ll \tilde{\nu}_2$ on $(E_2, \mathcal{E}_2)$) and regular conditional probability measures $\tilde{\mu}_{2|1} \ll \tilde{\nu}_{2|1}$ (resp. $\tilde{\mu}_{1|2} \ll \tilde{\nu}_{1|2}$). Recall the chain rule of relative entropy is

$$\mathrm{KL}(\tilde{\mu} \,||\, \tilde{\nu}) = \mathrm{KL}(\tilde{\mu}_1 \,||\, \tilde{\nu}_1) + \mathbb{E}_{\tilde{\mu}}\left[\log \frac{d\tilde{\mu}_{2|1}}{d\tilde{\nu}_{2|1}}\right] = \mathrm{KL}(\tilde{\mu}_2 \,||\, \tilde{\nu}_2) + \mathbb{E}_{\tilde{\mu}}\left[\log \frac{d\tilde{\mu}_{1|2}}{d\tilde{\nu}_{1|2}}\right] . \tag{A.2}$$

Observe that each of the terms in the equalities is non-negative In particular, when ψ is non-injective, points of $(E_1, \mathcal{E}_1)$ become equivalent; $(E_2, \mathcal{E}_2)$ is a compressed version, and the probability measures μ and ν are similarly compressed.

Proof of Lemma 3.4.2. Given probability measures on $(E_1, \mathcal{E}_1)$ $\mu \ll \nu$ (with density m such that $\mu = m \cdot \nu$) and a measurable map $\psi : (E_1, \mathcal{E}_1) \to (E_2, \mathcal{E}_2)$, construct the probability measure $\tilde{\mu}$ on $(E_1 \times E_2, \mathcal{E}_1 \otimes \mathcal{E}_2)$ as

$$\tilde{\mu}(A \times B) = \mu(A \cap \psi^{-1}B) = \int_A \mu(dx_1) \int_B \delta_{\psi(x_1)}(dx_2) \,, \quad A \in \mathcal{E}_1, \; B \in \mathcal{E}_2 \,,$$

and likewise for $\tilde{\nu}$. Then in the notation of (A.2), $\tilde{\mu}_1 = \mu \ll \nu = \tilde{\nu}_1$, and $\tilde{\mu}_{2|1} = \delta_{\psi(x_1)} = \tilde{\nu}_{2|1}$. Therefore,

$$\mathrm{KL}(\tilde{\mu} \mid\mid \tilde{\nu}) = \mathrm{KL}(\tilde{\mu}_1 \mid\mid \tilde{\nu}_1) = \mathrm{KL}(\mu \mid\mid \nu) \,. \tag{A.3}$$

Alternatively, $\tilde{\mu}_2 = \mu \circ \psi^{-1}$, $\tilde{\nu}_2 = \nu \circ \psi^{-1}$, and it is straightforward to show that

$$\mathbb{E}_{\tilde{\mu}}\left[\log \frac{d\tilde{\mu}_{1|2}}{d\tilde{\nu}_{1|2}}\right] = \mathbb{E}_{\tilde{\mu}}\left[\log \frac{d\tilde{\mu}_1}{d\tilde{\nu}_1}\right] - \mathbb{E}_{\tilde{\mu}}\left[\log \frac{d\tilde{\mu}_2}{d\tilde{\nu}_2}\right] = \mathbb{E}_{\mu}\left[\log \frac{m}{m \circ \psi}\right] = \Delta_{\psi}(\mu \mid\mid \nu) \geq 0. \,. \tag{A.4}$$

Therefore,

$$\mathrm{KL}(\tilde{\mu} \mid\mid \tilde{\nu}) = \mathrm{KL}(\mu \mid\mid \nu) = \mathrm{KL}(\mu \circ \psi^{-1} \mid\mid \nu \circ \psi^{-1}) + \Delta_{\psi}(\mu \mid\mid \nu) \,. \tag{A.5}$$

□

A.1.1 Proof of Theorem 3.4.1

Theorem 3.4.1. *Assume that $P_{\mathcal{D}}$ is $\mathcal{G}$-invariant. Then Theorem 3.2.1 holds with $\widehat{R}^{\circ}_{\ell}(Q, \mathcal{D}^n)$ as in (3.4) substituted for $\widehat{R}_{\ell}(Q, \mathcal{D}^n)$.*

The proof of our PAC-Bayes bound for data augmentation makes use of the following result due to Lever et al. [2013].

Theorem A.1.1 (Lever et al. [2013], Theorem 1)**.** *For any functions $A(f, \mathcal{D}^n)$, $B(f)$ over F, either of which may be a statistic of the training data $\mathcal{D}^n$, any distribution P over F, any $\delta \in (0, 1]$, any $t > 0$, and a convex function $\mathscr{D} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$, with probability $P_{\mathcal{D}}^n$ at least $1 - \delta$, for all distributions Q on F,*

$$\mathscr{D}\Big(\mathbb{E}_{f\sim Q}[A(f, \mathcal{D}^n)], \mathbb{E}_{f\sim Q}[B(f)]\Big) \leq \frac{1}{t}\left(\mathrm{KL}(Q \mid\mid P) + \log \frac{\mathcal{L}_P}{\delta}\right) , \tag{A.6}$$

where $\mathcal{L}_P := \mathbb{E}_{\mathcal{D}^n\sim P_{\mathcal{D}}, f\sim P}[e^{t\mathscr{D}(A(f,\mathcal{D}^n), B(f))}]$ is the Laplace transform of $\mathscr{D}(A(f), B(f))$.

As Lever et al. [2013] discuss, many PAC-Bayes bounds in the literature can be obtained as special cases of Lemma A.1.1, including Catoni's bound in

Theorem 3.2.1. In that case, which applies to 0-1 loss, $t = n$, $A(f) = \widehat{R}_\ell(f, \mathcal{D}^n)$, $B(f) = R_\ell(f)$, $C = n^{-1}$ and

$$\mathscr{D}_C(q,p) := -\log(1 - p(1 - e^{-C})) - Cq\,, \quad q,p \in (0,1),\ C > 0 \tag{A.7}$$

$$= -\log \mathbb{E}_{Z\sim\mathrm{Bern}(p)}[e^{-CZ}] - Cq\,. \tag{A.8}$$

Basic calculations show that with these quantities, $\mathcal{L}_P = 1$.

$$\mathcal{L}_P = \mathbb{E}_{\mathcal{D}^n,f}[e^{t\mathscr{D}(A(f;\mathcal{D}^n),B(f))}] = \mathbb{E}_{\mathcal{D}^n,f}[e^{t\mathscr{D}(\widehat{R}_\ell(f;\mathcal{D}^n),R_\ell(f))}] \tag{A.9}$$

$$= \mathbb{E}_{f,\mathcal{D}^n}[e^{t(-\log \mathbb{E}[e^{CZ}] - C\widehat{R}_\ell(f;\mathcal{D}^n))}] \tag{A.10}$$

$$= \mathbb{E}_f[e^{-t\log \mathbb{E}[e^{-CZ}]}\mathbb{E}_{\mathcal{D}^n}[e^{-tC\widehat{R}_\ell(f;\mathcal{D}^n)}]] \tag{A.11}$$

$$= \mathbb{E}_f[e^{-\log \mathbb{E}[e^{-CZ}]}\mathbb{E}_{Z_{1:n}\sim\mathrm{Bern}(R_\ell(f))}[e^{-tC\frac{1}{n}\sum_{i=1}^n Z_i}]] \tag{A.12}$$

$$= \mathbb{E}_f[e^{-\log \mathbb{E}[e^{-CZ}]}\Pi_{i=1}^n\mathbb{E}_{Z_i\sim\mathrm{Bern}(R_\ell(f))}[e^{-\frac{1}{n}Z_i}]] \tag{A.13}$$

$$= \mathbb{E}_f[e^{-t\log \mathbb{E}[e^{-CZ}]}(\mathbb{E}_{Z\sim\mathrm{Bern}(R_\ell(f))}[e^{-CZ}])^n] \tag{A.14}$$

$$= \mathbb{E}_f[e^{-t\log \mathbb{E}_{Z\sim\mathrm{Bern}(R_\ell(f))}[e^{-CZ}]}e^{t\log \mathbb{E}_{Z\sim\mathrm{Bern}(R_\ell(f))}[e^{-CZ}]}] \tag{A.15}$$

$$= \mathbb{E}_f[1] = 1 \tag{A.16}$$

Recall that

$$\widehat{R}_\ell(f,\mathcal{D}^n) := \frac{1}{n}\sum_{i=1}^n \ell(f(X_i),Y_i) \tag{A.17}$$

$$\widehat{R}^\circ_\ell(f,\mathcal{D}^n) := \frac{1}{n}\sum_{i=1}^n \mathbb{E}_{G\sim\lambda}[\ell(f(GX_i),Y_i)] \tag{A.18}$$

$$\widehat{\widehat{R}^\circ_\ell}(f,\mathcal{D}^n) := \frac{1}{nm}\sum_{i=1}^n\sum_{j=1}^m \ell(f(G_{ij}X_i),Y_i)\,. \tag{A.19}$$

Let (G_{ij}) denote the collection of $m \cdot n$ random augmentation transformations sampled i.i.d. from λ.

Lemma A.1.2. *Let ℓ be the binary loss, P any distribution on F, and assume that $P_\mathcal{D}$ is $\mathcal{G}$-invariant. Then*

$$\mathbb{E}_{f\sim P}\Big[\mathbb{E}_{\mathcal{D}^n\sim P_\mathcal{D}}\Big[e^{n\mathscr{D}_C(\widehat{R}^\circ_\ell(f,\mathcal{D}^n),R_\ell(f))}\Big]\Big] \leq \mathbb{E}_{f\sim P}\Big[\mathbb{E}_{\mathcal{D}^n\sim P_\mathcal{D}}[e^{n\mathscr{D}_C(\widehat{R}_\ell(f,\mathcal{D}^n),R_\ell(f))}]\Big] = 1 \tag{A.20}$$

and

$$\mathbb{E}_{f\sim P}\Big[\mathbb{E}_{\mathcal{D}^n\sim P_{\mathcal{D}}}\Big[e^{n\mathscr{D}_C(\widehat{R}^{\circ}_{\ell}(f,\mathcal{D}^n),R_\ell(f))}\Big]\Big] \leq \mathbb{E}_{f\sim P}\Big[\mathbb{E}_{\mathcal{D}^n\sim P_{\mathcal{D}}}[e^{n\mathscr{D}_C(\widehat{R}_\ell(f,\mathcal{D}^n),R_\ell(f))}]\Big] = 1\,. \tag{A.21}$$

Proof. Since the observations (X_i, Y_i) are i.i.d., the expectation over $\mathcal{D}^n$ on the left-hand side of (A.20) requires evaluating $\mathbb{E}_{\mathcal{D}^n\sim P_{\mathcal{D}}}\Big[e^{-C\mathbb{E}_{G\sim\lambda}[\ell(f(X_i),Y_i))]}\Big]$. Using the convexity of e^{-x}, Jensen's inequality and Fubini's theorem yield

$$\begin{aligned}\mathbb{E}_{(X_i,Y_i)\sim P_{\mathcal{D}}}\Big[e^{-C\mathbb{E}_{G\sim\lambda}[\ell(f(GX_i),Y_i))]}\Big] &\leq \mathbb{E}_{(X_i,Y_i)\sim P_{\mathcal{D}}}\Big[\mathbb{E}_{G\sim\lambda}\Big[e^{-C\ell(f(GX_i),Y_i))}\Big]\Big] \\ &= \mathbb{E}_{G\sim\lambda}\Big[\mathbb{E}_{(X_i,Y_i)\sim P_{\mathcal{D}}}\Big[e^{-C\ell(f(GX_i),Y_i))}\Big]\Big]\,.\end{aligned} \tag{A.22}$$

Now, $\mathcal{G}$-invariance of $P_{\mathcal{D}}$ implies that $\mathbb{E}_{(X_i,Y_i)\sim P_{\mathcal{D}}}[h(gX_i,Y_i)] = \mathbb{E}_{(X_i,Y_i)\sim P_{\mathcal{D}}}[h(X_i,Y_i)]$ for all measurable functions $h : \mathcal{X}\times\mathcal{Y}\to\mathbb{R}_+$ and all $g\in\mathcal{G}$, which extends to independent random G by Fubini's theorem. Therefore,

$$\mathbb{E}_{G\sim\lambda}\Big[\mathbb{E}_{(X_i,Y_i)\sim P_{\mathcal{D}}}\Big[e^{-C\ell(f(GX_i),Y_i))}\Big]\Big] = \mathbb{E}_{(X_i,Y_i)\sim P_{\mathcal{D}}}\Big[e^{-C\ell(f(X_i),Y_i))}\Big] = \mathbb{E}_{Z\sim\mathrm{Bern}(R_\ell(f))}[e^{-CZ}]\,,$$

which implies (A.20).

For the second inequality (A.21), observe that by Jensen's inequality,

$$\begin{aligned}\mathbb{E}_{\mathcal{D}^n\sim P_{\mathcal{D}}}\Big[e^{-nC\widehat{R}^{\circ}_{\ell}(f,\mathcal{D}^n)}\Big] &= \prod_{i=1}^n \mathbb{E}_{(X_i,Y_i)\sim P_{\mathcal{D}}}\left[\mathbb{E}_{(G_{ij})_{j=1}^m\sim\lambda}\left[\exp\left(-\frac{C}{m}\sum_{j=1}^m \ell(f(G_{ij}X_i),Y_i)\right)\right]\right] \\ &\leq \prod_{i=1}^n \mathbb{E}_{(X_i,Y_i)\sim P_{\mathcal{D}}}\left[\mathbb{E}_{(G_{ij})_{j=1}^m\sim\lambda}\left[\frac{1}{m}\sum_{j=1}^m e^{-C\ell(f(G_{ij}X_i),Y_i)}\right]\right] \\ &= \prod_{i=1}^n \mathbb{E}_{(X_i,Y_i)\sim P_{\mathcal{D}}}\Big[\mathbb{E}_{G\sim\lambda}\Big[e^{-C\ell(f(GX_i),Y_i)}\Big]\Big]\end{aligned}$$

Using the $\mathcal{G}$-invariance of $P_{\mathcal{D}}$ once again, we have

$$\mathbb{E}_{\mathcal{D}^n\sim P_{\mathcal{D}}}\Big[e^{-nC\widehat{R}^{\circ}_{\ell}(f,\mathcal{D}^n)}\Big] \leq \mathbb{E}_{\mathcal{D}^n\sim P_{\mathcal{D}}}\Big[e^{-nC\widehat{R}_\ell(f,\mathcal{D}^n)}\Big] = \Big(\mathbb{E}_{Z\sim\mathrm{Bern}(R_\ell(f))}[e^{-CZ}]\Big)^n\,,$$

which implies (A.21). □

Note that Lemma A.1.2 depends on the observation that for any distribution over functions, the symmetrized risk will be a lower-variance estimator than the empirical risk but will remain unbiased. This is only the case for *invariant* data-generating distributions – symmetrizing the risk of a non-symmetric $P_{\mathcal{D}}$ will not necessarily result in a valid PAC-Bayes bound.

A.2 Examples, counterexamples, tighter bounds

A.2.1 Counterexamples

Feature averaging and non-convex losses. We consider the binary classification setting with the zero-one loss and some function class f bounded in $[0, 1]$ – that is $\ell(x, y) = \mathbb{1}[|f(x) - y| > 1]$. Suppose that there exists some invariance $\mathcal{G}$ in the data such that $y(x) = y(gx)$ for all x, g. Then consider a function which, for some small ϵ, outputs $f(x) = \frac{1}{2} + y\epsilon$ on a $1 - 2\epsilon$ fraction of each equivalence class of the inputs, and $1 - y$ on 2ϵ of the inputs in each equivalence class. Then $\mathbb{E}[f(gx)] = (1 - 2\epsilon)(\frac{1}{2} + y\epsilon) + 2\epsilon(1 - y)$. When $y = 0$, this expectation is $\frac{1}{2} + \epsilon$, and when $y = 1$ it is $\frac{1}{2}[1 - \epsilon - 2\epsilon^2] < \frac{1}{2}$, so the feature-averaged model would have risk 1 whereas the original model had risk 0.

Non-uniform data-generating distributions. When the data-generating distribution is not uniform over the set $\mathcal{T}$, then performing data augmentation with $\mathcal{T}$ will not necessarily lead to a more accurate estimate of the model's empirical risk. For example, consider the task of learning a function g satisfying $g(x) = g(-x)$, bounded in magnitude by some constant A. Suppose, however, that positive numbers are much more likely under the data generating distribution, with $p(\mathbb{R}^+) = 1 - \epsilon$ for small ϵ. Then the function $f(x) = \mathbb{1}[x > 0]g(x)$ will satisfy $\mathbb{E}[\|f(X_S) - g(X_s)\|] \neq \mathbb{E}[\|f(X_{S^{\text{aug}}}) - g(X_{S^{\text{aug}}})\|]$. So the augmented risk is no longer an unbiased estimator of the empirical risk. Further, in this particular case its variance is also higher, as it will be equal to $\frac{1}{2}\text{Var}(g(x))$, in contrast to $\epsilon\text{Var}(g(x))$.

A.2.2 Tighter PAC-Bayes bound for data augmentation

Although Theorem 3.4.1 establishes that the i.i.d. PAC-Bayes bound (3.6) is valid for exact DA, the proof of Theorem 3.4.1 indicates that a tighter bound is possible. In particular, recall that when $P_{\mathcal{D}}$ is $\mathcal{G}$-invariant[Bloem-Reddy and Teh, 2019, Chen et al., 2019],

$$\mathbb{E}_{G\sim\lambda}[\ell(f(GX), Y)] = \mathbb{E}_{(X,Y)\sim P_{\mathcal{D}}}[\ell(f(X), Y) \mid \Phi] := \ell_f^\circ(\Phi) \ .$$

$\ell_f^\circ(\Phi)$ is a random variable, the average loss on the random orbit with representative Φ, whose distribution is induced by $P_{\mathcal{D}}$. Therefore, we can write $\mathcal{L}_P$ in (A.6) as

$$\mathcal{L}_P = \mathbb{E}_{f\sim P}\left[\left(\frac{\mathbb{E}_{\Phi\sim P_{\mathcal{D}}}\left[e^{-C\ell_f^\circ(\Phi)}\right]}{\mathbb{E}_{Z\sim \mathrm{Bern}(R_\ell(f))}\left[e^{-CZ}\right]}\right)^n\right] \leq 1 \ .$$

In general, this cannot be computed in closed form. However, it might be possible to estimate using the data (with appropriate modifications to the resulting bound) and samples $f \sim P$.

A.3 Computation details for PAC-Bayes bounds

PAC-Bayes bounds for neural networks are computed via the following procedure: a deterministic neural network is trained to minimize the cross-entropy loss on the dataset. After it has reached a suitable training accuracy, we use these parameters as the initialization for the means and variances of the stochastic neural network weights used for the PAC-Bayes bounds. We directly optimize a surrogate of the PAC-Bayes bound (using the cross-entropy loss instead of the zero-one accuracy and using the reparameterization trick to get the derivatives of the variance parameters). The exact computation of the PAC-Bayes bound uses the union bound and discretization of the PAC-Bayes prior as described in [Dziugaite and Roy, 2017]. Reported values are at optimization convergence.

The experiment code is provided with the paper submission, but we describe here at a high level the different models used in our empirical evaluations.

FashionMNIST CNN: the convolutional network used for FashionMNIST consists of two convolutional layers (with batch norm and max pooling) followed by a single fully connected layer.

LiDAR Permutation-Invariant Network: we use a scaled-down version of the PointNet architecture [Qi et al., 2017]. We include two layers of 1D convolutions followed by a max-pooling layer that selects the maximum over input points for each channel. This layer is followed by two fully-connected layers leading into the final output.

Partially-Invariant Network: we alter the previous architecture slightly so that it is only invariant to *subgroups* of the permutation group on its inputs. Specifically, we partition the input into 8 disjoint subsets, and apply the previous model's permutation-invariant embedding layers to each partition. The result is a feature representation that is invariant to permutations within each partition of the input, but not between partitions. This representation is then fed through the same architecture. We note that we keep the number of convolutional filters per layer constant, which results in a larger feature embedding by a factor of 8 that is fed into the first fully connected layer. As a result, this model has significantly more parameters than the fully permutation-invariant model.

Fully Connected Network: the max-pooling operator of the previous two architectures is omitted. This network has many more parameters than either of the first two models, and is not invariant to any subgroup of the permutation group.

B

Training speed and model selection

B.1 Linear model combination

The estimator $\mathcal{L}(\mathcal{D})$ reveals an intriguing connection between pruning in linear model combinations and Bayesian model selection. This connection arises when we consider the weight assigned to a model in a linear model combination when models are fit on data points iteratively, as in the marginal likelihood estimation algorithms discussed previously.

We study a setting that presents a bridge between Bayesian updating and optimization by considering the limiting behaviour of a linear regressor trained on predictions output by Bayesian models fitted iteratively as seen in Algorithm 4.3.1. Concretely, we treat samples from the model posterior $P(Y_i|\mathcal{D}_{<i}, X_i, \mathcal{M}_j)$ as entries $\Phi[i, j]$ in a design matrix Φ, and then consider properties of an optimal linear predictor trained on Φ. While this does not perfectly replicate the setting of deep neural networks trained with gradient descent, it highlights an important property of gradient descent on non-stationary features: the feature which contributes the most to the output of a linear ensemble will often be the one which was most predictive of the target *on average* over the course of training, rather than the one which is most correlated with the target at the end of training.

Our analysis requires a number of assumptions to be formalized. We assume a data set $\mathcal{D} = (X_i, Y_i)_{i=1}^n$ and a collection of k models $\mathcal{M}_1, \ldots, \mathcal{M}_k$. We train a linear regressor w to fit the posterior predictive distributions of the models to the target Y_i; i.e. to regress on the dataset

$$(\Phi, Y) = \left(\phi_i = (\hat{Y}_1^i, \ldots, \hat{Y}_n^i), Y_i\right)_{i=1}^n \text{ with } \hat{Y}_j^i \sim P(\hat{Y}|\mathcal{D}_{<i}, X_i, \mathcal{M}_j). \tag{B.1}$$

The following result shows that the optimal linear regressor on this data generating distribution assigns the highest weight to the model with the highest $\mathcal{L}(\mathcal{D})$ whenever the model errors are independent. This shows that magnitude pruning in a linear model combination is equivalent to approximate Bayesian model selection, under certain assumptions on the models.

Proposition B.1.1. *Let $\mathcal{M}_1, \ldots, \mathcal{M}_k$ be Bayesian linear regression models with fixed noise variance σ_N^2 and Gaussian likelihoods. Let Φ be a (random) matrix of posterior prediction samples, of the form $\Phi[i,j] = \hat{y}_i^j \sim P(y_j|\mathcal{D}_{<j}, x_j, \mathcal{M}_i)$. Suppose the following two conditions on the columns of Φ are satisfied: $\mathbb{E}\langle\Phi[:,i], y\rangle = \mathbb{E}\langle\Phi[:,j], y\rangle$ for all i, j, and $\mathbb{E}\langle\Pi_{y^\perp}\phi_i, \Pi_{y^\perp}\phi_j\rangle = 0$. Let w^* denote the least-squares solution to the regression problem $\min_w \mathbb{E}_\Phi\|\Phi w - y\|^2$. Then the following holds*

$$\text{argmax}_i w_i^* = \text{argmax}_i \mathcal{L}(\mathcal{D}|\mathcal{M}_i) \qquad \forall w^* = \text{argmin}_w \mathbb{E}\|\Phi w - y\|^2 \,. \tag{B.2}$$

The assumption on the independence of model errors is crucial in the proof of this result: families of models with large and complementary systematic biases may not exhibit this behaviour. We observe in Section 4.3.3 that the conditions of Proposition 1 are approximately satisfied in a variety of model comparison problems, and running SGD on a linear combination of Bayesian models still leads to solutions that approximate Bayesian model selection. We conjecture that analogous phenomena occur during training within a neural network. The proof of Proposition B.1.1 depends on the observation that, given a collection of features, the best least-squares predictor will assign the greatest weight to the feature that best predicts the training data. While neural networks are not linear ensembles of fixed models, we conjecture that, especially for later layers of the network, a similar

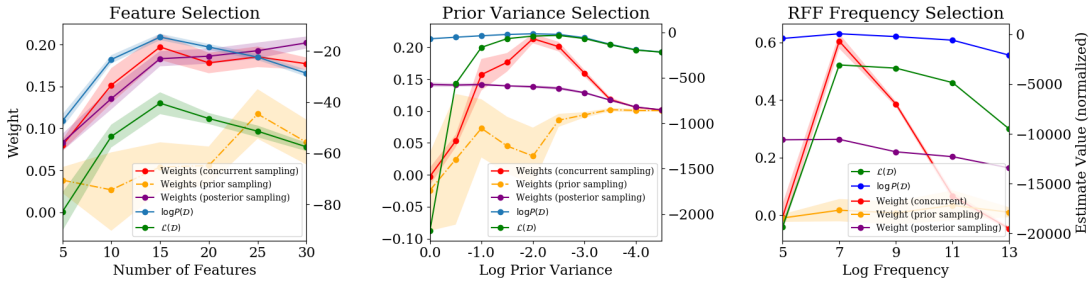


Figure B.1: Relative rankings given by optimize-then-prune, ML, and estimated $\mathcal{L}(\mathcal{D})$. Left: feature selection. Middle: prior variance selection. Right: RFF frequency selection. Rankings are consistent with what our theoretical results predict. Results are averaged over 5 runs.

phenomenon will occur wherein weights from nodes that are more predictive of the target values over the course of training will be assigned higher magnitudes. We empirically investigate this hypothesis in Section 4.4.2.

We consider three model selection problems in our empirical evaluations. In **prior variance selection** we evaluate a set of BLR models on a synthetic linear regression data set. Each model $\mathcal{M}_i$ has a prior distribution over the d parameters of the form $w \sim \mathcal{N}(0, \sigma_i^2 I_d)$ for some σ_i^2, and the goal is to select the optimal prior variance (in other words, the optimal regularization coefficient). We additionally evaluate an analogous initialization variance selection method on an NTK network trained on a toy regression dataset. In **frequency (lengthscale) selection** we use as input a subset of the handwritten digits dataset MNIST given by all inputs labeled with a 0 or a 1. We compute random Fourier features (RFF) of the input to obtain the features for a Bayesian linear regression model, and perform model selection over the frequency of the features (full details on this in the appendix). This is equivalent to obtaining the lengthscale of an approximate radial basis function kernel. In **feature dimension selection**, we use a synthetic dataset [Wilson and Izmailov, 2020] of the form $(\mathbf{X}, \mathbf{y})$, where $x_i = (y_i + \epsilon_1, y_i + \ldots, y_i + \epsilon_{15}, \epsilon_{16}, \ldots, \epsilon_{30})$ with ϵ_i i.i.d. noise variables. We then consider a set of models $\{\mathcal{M}_k\}$ with feature embeddings $\phi_k(x_i) = x_i[1, \ldots, k]$. The optimal model in this setting is the one which uses exactly the set of 'informative' features $x[1, \ldots, 15]$.

We empirically evaluate the claims of Proposition B.1.1 in settings where the assumptions only approximately hold. Concretely, we consider learning problems where the outputs of the models have roughly equal norm and approximately independent errors. We compare the ranking given by the true log marginal likelihood, the estimated $\mathcal{L}(\mathcal{D})$, and the weight assigned to each model by the trained linear regressor. We consider three variations on how sampled predictions from each model are drawn to generate the features ϕ_i: sampling the prediction for point $\hat{Y}_i$ from $P(\hat{Y}_i|\mathcal{D}_{<i})$ ('concurrent sampling' – this is the setting of Proposition B.1.1), as well as two baselines: the posterior $P(\hat{Y}_i|\mathcal{D})$ ('posterior sampling'), and the prior $P(\hat{Y}_i)$ ('prior sampling'). The baselines illustrate the importance of *concurrently* fitting the models and the linear weights; fitting linear weights over a set of trained models simply identifies the one with the best training set performance, and fitting linear weights to the untrained models tends to favour those with lower-magnitude predictions. Their inclusion further highlights that in the model selection problems on which we evaluate the methods, it is not the case that either prior or posterior predictive likelihood is sufficient to correctly identify the best model.

Concretely, we find that the rankings of the marginal likelihood, its lower bound, and of the ranking given by concurrent optimization agree on the best model in all three of the model selection problems outlined previously. The prior and posterior sampling procedure baselines do not exhibit a consistent ranking with the log ML, and indeed exhibit opposite trends in the prior variance selection task. We visualize these results for the feature dimension selection problem in Figure B.1.

Subnetwork selection in neural networks

Finally, we evaluate whether our previous insights apply to submodels within a neural network, suggesting a potential mechanism which may bias SGD towards parameters with better generalization performance. Based on the previous experiments, we expect that nodes that have a lower sum over training errors (if evaluated as a classifier on their own) are favoured by gradient descent and therefore have a larger final weight than those which are less predictive of the data. If so, we can then view

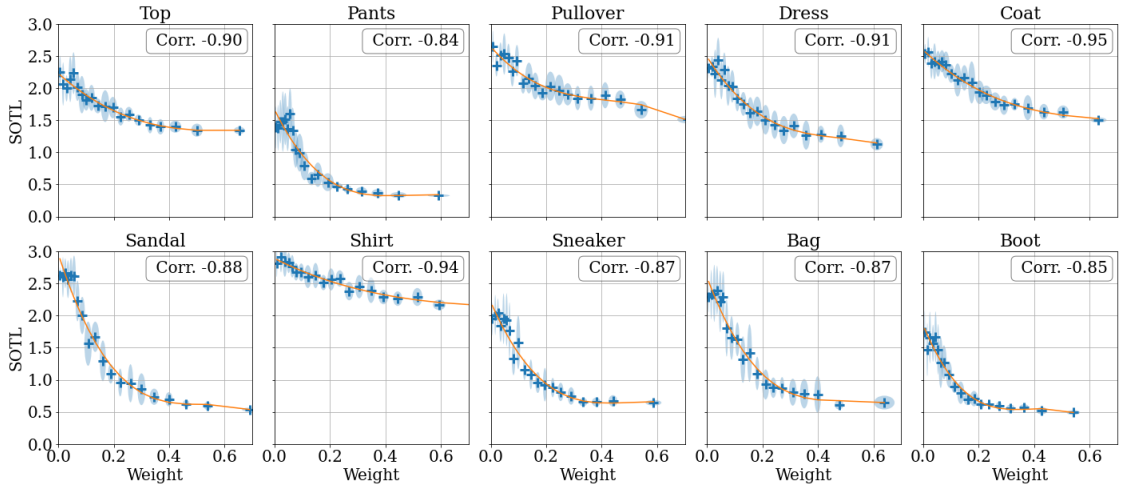


Figure B.2: Weight assigned to subnetwork by SGD in a deep neural network (x-axis) versus the subnetwork performance (estimated by the sum of cross-entropy, on the y-axis) for different FashionMNIST classes. The light blue ovals denote depict 95% confidence intervals, estimated over 10 seeds (i.e. 2σ for both the weight and SOTL). The orange line depicts the general trend.

SGD followed by pruning (in the final linear layer of the network) as performing an approximation of a Bayesian model selection procedure. We replicate the model selection problem of the previous setting, but replace the individual models with the activations of the penultimate layer of a neural network, and replace the linear ensemble with the final linear layer of the network. Full details on the experimental set-up can be found in Appendix B.3.3. We find that our hypotheses hold here: SGD assigns larger weights to subnetworks that perform well, as can be seen in Figures B.2 and B.3. This suggests that SGD is biased towards functions that generalize well, even within a single neural network. We find the same trend holds for CIFAR-10, which is shown in Appendix B.3.3.

B.2 Proofs of theoretical results

Proposition 4.3.1. *Both $\hat{\mathcal{L}}$ and $\hat{\mathcal{L}}_k$ as defined in Equation 4.4 are estimators of lower bounds on the log marginal likelihood; that is*

$$\mathbb{E}[\hat{\mathcal{L}}(\mathcal{D})] = \mathcal{L}(\mathcal{D}) \leq \log P(\mathcal{D}) \quad \textit{and} \quad \mathbb{E}[\hat{\mathcal{L}}_k(\mathcal{D})] = \mathcal{L}_k(\mathcal{D}) \leq \log P(\mathcal{D}) \,. \tag{4.5}$$

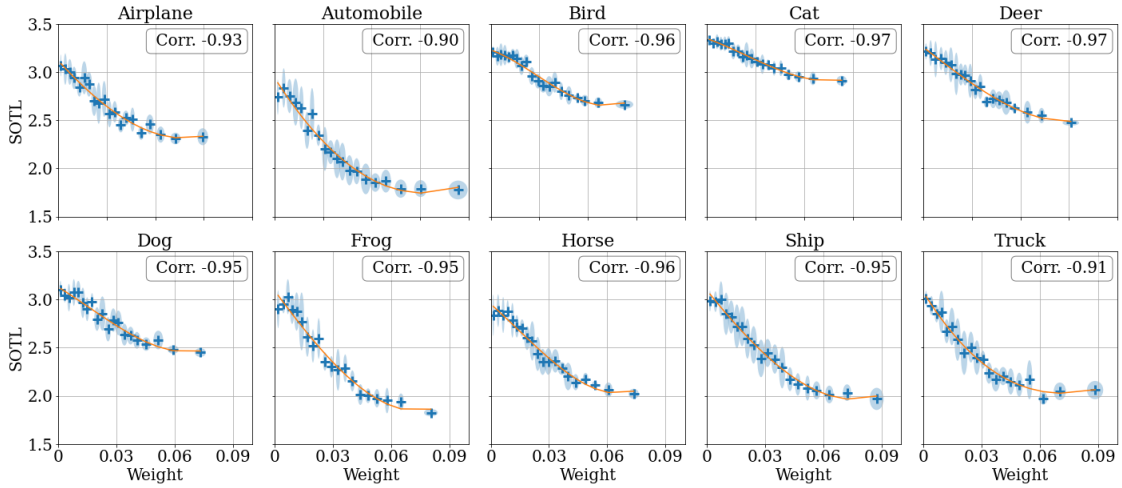


Figure B.3: Weight assigned to subnetwork by SGD in a deep neural network (x-axis) versus the subnetwork performance (estimated by the sum of cross-entropy, on the y-axis) for different CIFAR-10 classes. The light blue ovals denote depict 95% confidence intervals, estimated over 10 seeds (i.e. 2σ for both the weight and SOTL). The orange line depicts the general trend.

Further, the bias term in $\mathcal{L}$ can be quantified as follows.

$$\mathcal{L}(\mathcal{D}) = \log P(\mathcal{D}) - \sum_{i=1}^{n} \mathrm{KL}((P(\theta|\mathcal{D}_{<i}) \;||\; P(\theta|\mathcal{D}_{<i+1})) \tag{4.6}$$

Proof. The result for $\mathcal{L}$ follows from a straightforward derivation:

$$\begin{aligned}
\mathcal{L}(\mathcal{D}) &= \sum \int \log P(\mathcal{D}_i|\theta) dP(\theta|\mathcal{D}_{<i}) && \text{(B.3)}\\
&= \sum \int \log[\frac{P(\mathcal{D}_i|\theta)P(\theta|\mathcal{D}_{<i})P(\mathcal{D}_i|\mathcal{D}_{<i})}{P(\theta|\mathcal{D}_{<i})P(\mathcal{D}_i|\mathcal{D}_{<i})}] dP(\theta|\mathcal{D}_{<i}) && \text{(B.4)}\\
&= \sum \int \log \frac{P(\theta|\mathcal{D}_{\leq i}))}{P(\theta|\mathcal{D}_{<i})} dP(\theta|\mathcal{D}_{<i}) + \sum \log P(\mathcal{D}_i|\mathcal{D}_{<i}) && \text{(B.5)}\\
&= \sum \left(\log P(\mathcal{D}_i|\mathcal{D}_{<i}) - \mathrm{KL}((\;||\;P)(\theta|\mathcal{D}_{<i})||P(\theta|\mathcal{D}_{\leq i})) \right) && \text{(B.6)}\\
&= \log P(\mathcal{D}) - \sum_{i=1}^{n} \mathrm{KL}((\;||\;P)(\theta|\mathcal{D}_{<i})||P(\theta|\mathcal{D}_{\leq i})). && \text{(B.7)}
\end{aligned}$$

The result for $\hat{\mathcal{L}}_k$ follows immediately from Jensen's inequality, yielding

$$\sum \mathbb{E}[\log \sum_{j=1}^{k} \frac{1}{k} p(\mathcal{D}_i|\theta_j)] \leq \sum \log \mathbb{E}[\sum_{j=1}^{k} \frac{1}{k} p(\mathcal{D}_i|\theta_j)] = \sum \log \mathbb{E}[p(\mathcal{D}_i|\theta_j)] = \log P(\mathcal{D}) \; . \tag{B.8}$$

Because $\mathcal{L}_k$ applies Jensen's inequality to a random variable with decreasing variance as a function of k, we expect the bias of $\mathcal{L}_k$ to decrease as k grows, an observation characterized in Section 4.3.3. □

Proposition 4.3.2. *Let $P(Y_i|\mathcal{D}_{<i}, X_i) = \mathcal{N}(\mu_i, \sigma_i^2)$ for some μ_i, σ_i^2. Define the standard mean and variance estimators $\hat{\mu}_i = \frac{1}{N}\sum_{j=1}^{N} f(\theta_j^i, x_i)$ and $\hat{\sigma}_i^2 = \frac{1}{N-1}\sum(f(\theta_j^i, x_i) - \hat{\mu})^2$. Then the estimator*

$$\hat{\mathcal{L}}_S(\mathcal{D}) = \sum_{i=1}^{n} \log P(Y_i|\hat{\mu}_i, \hat{\sigma}_i^2) \tag{4.7}$$

is a lower bound on the log ML: i.e. $\mathbb{E}[\hat{\mathcal{L}}_S(\mathcal{D})] \leq \log P(\mathcal{D})$.

Proof. To show that the sum of the estimated log likelihoods is a lower bound on the log marginal likelihood, it suffices to show that each term in the sum of the estimates is a lower bound on the corresponding term in log marginal likelihood expression. Thus, without loss of generality we consider a single data point $\mathcal{D}_i = (x, y)$ and posterior distribution $p(y|x, \mathcal{D}_{<i}) = \mathcal{N}(\mu, \sigma^2)$.

Let $y \in \mathbb{R}$, $\hat{\mu}, \hat{\sigma}$ the standard estimators for sample mean and variance given sample $\hat{Y} \in \mathbb{R}^k$ sampled from $\mathcal{N}(\mu, \sigma^2)$. We want to show

$$\mathbb{E}_{\hat{Y}\sim\mathcal{N}(\mu,\sigma^2)}[\ln p(y|\hat{\mu}, \hat{\sigma}^2)] \leq \ln p(y|\mu, \sigma^2). \tag{B.9}$$

We first note that $\hat{\mu}(\hat{Y}) \perp \hat{\sigma}(\hat{Y})$ for $\hat{Y}$ a collection of i.i.d. Gaussian random variables [Basu, 1955]. We also take advantage of the fact that the log likelihood of a Gaussian is concave with respect to its μ parameter and its σ^2 parameter. Notably, the log likelihood is *not* concave w.r.t. the joint pair (μ, σ^2), but because the our estimators are independent, this will not be a problem for us. We proceed as follows by first decomposing the expectation over the samples $\hat{Y}$ into an expectation over $\hat{\mu}$ and $\widehat{\sigma^2}$

$$\begin{aligned}\mathbb{E}_{X\sim\mathcal{N}(\mu,\sigma^2)}[\ln p(y|\hat{\mu}, \hat{\sigma}^2)] &= \mathbb{E}_{\hat{\mu},Y_2,\ldots,Y_N} \ln p(y|\hat{\mu}, \hat{\sigma}^2) &\text{(B.10)}\\ &= \mathbb{E}_{\hat{\mu}}\mathbb{E}_{\hat{\sigma}^2} \ln p(y|\hat{\mu}, \hat{\sigma}^2) &\text{(B.11)}\end{aligned}$$

We apply Jensen's inequality first to the inner expectation, then to the outer.

$$\begin{aligned}&\leq \mathbb{E}_{\hat{\mu}} \ln p(y|\hat{\mu}, \mathbb{E}[\hat{\sigma}^2]) = \mathbb{E}_{\hat{\mu}} \ln p(y|\hat{\mu}, \sigma^2) &\text{(B.12)}\\ &\leq \ln p(y|\mu, \sigma^2) &\text{(B.13)}\end{aligned}$$

So we obtain our lower bound. □

Theorem 4.3.3. *Let $\mathcal{D} = (X_i, Y_i)_{i=1}^n$ and let $(\theta_j^i)_{i,j=1}^{n,J}$ be generated by the procedure outlined in Algorithm 4.3.1. Then the estimators $\hat{\mathcal{L}}, \hat{\mathcal{L}}_S$, and $\hat{\mathcal{L}}_k$, applied to the collection (θ_j^i), are lower bounds on $\log P(\mathcal{D})$. Further, expressing $-\log P(\mathcal{D}_i|\theta)$ as the ℓ_2 regression loss plus a constant, we then obtain*

$$\log P(\mathcal{D}) \geq \sum_{i=1}^{n} \mathbb{E}_{\theta_i \sim P(\cdot|\mathcal{D}_{<i})}[\log P(\mathcal{D}_i|\theta_i)] = \mathbb{E} \sum_{i=1}^{n} -\ell_2(\mathcal{D}_i, \theta_i) + c = \mathcal{L}(\mathcal{D}) \tag{4.8}$$

Proof. The heavy lifting for this result has largely been achieved by Propositions 4.3.1 and 4.3.2, which state that provided the samples θ_j^i are distributed according to the posterior, the inequalities will hold. It therefore remains only to show that the sample-then-optimize procedure yields samples from the posterior. The proof of this result can be found in Lemma 3.8 of Osband et al. [2018], who show that the optimum for the gradient descent procedure described in Algorithm 4.3.1 does indeed correspond to the posterior distribution for each subset $\mathcal{D}_{<i}$.

Finally, it is straightforward to express the lower bound estimator $\hat{\mathcal{L}}$ as the sum of regression losses. We obtain this result by showing that the inequality holds for each term $\log P(\mathcal{D}_i|\theta_i)$ in the summation.

$$\log P(\mathcal{D}_i|\theta) = \log[\exp\left(-\frac{(\theta^\top x_i - y_i)^2}{2\sigma^2}\right)\frac{1}{\sqrt{2\pi}\sigma}] \tag{B.14}$$

$$= -\frac{(\theta^\top x_i - y_i)^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2) \tag{B.15}$$

$$= c_1 \ell_2(\mathcal{D}_i, \theta) + c_2 \tag{B.16}$$

We note that in practice, the solutions found by gradient descent for finite step size and finite number of steps will not necessarily correspond to the exact local optimum. However, it is straightforward to bound the error obtained from this approximate sampling in terms of the distance of θ from the optimum θ^*. Denoting

the difference $|\theta - \theta^*|$ by δ, we get

$$|\log P(\mathcal{D}_i|\theta^*) - \log P(\mathcal{D}_i|\theta)| = |\frac{((\theta^*)^\top x_i - y_i)^2}{2\sigma^2} - \frac{((\theta)^\top x_i - y_i)^2}{2\sigma^2}| \tag{B.17}$$

$$\leq \frac{1}{2\sigma^2}|(\theta^*)^\top x_i - \theta^\top x_i|^2 \tag{B.18}$$

$$\leq |((\theta^*)^\top x_i)^2 - (\theta^\top x_i)^2| + |2y||\theta^\top x - (\theta^*)^\top x| \tag{B.19}$$

$$\leq |(\theta^* - \theta)^\top x + 2((\theta^*)^\top x)((\theta^* - \theta)^\top x)| + |2y||\theta^\top x - (\theta^*)^\top x| \tag{B.20}$$

$$\leq |\theta^* - \theta||x| + 2|\theta^* x||\theta^* - \theta||x| + |2y||x||\theta - \theta^*| \tag{B.21}$$

and so the error in the estimate of $\log P(\mathcal{D}|\theta)$ will be proportional to the distance $|\theta - \theta^*|$ induced by the approximate optimization procedure. □

Corollary 4.3.4. *Let $\mathcal{D}^n$ be a dataset indexed by our standard notation. Let f_0 be sampled from an infinitely wide neural network architecture $\mathcal{F}$ under some initialization distribution, and let f^i_∞ be the limiting solution under the training dynamics defined by He et al. [2020] applied to the initialization f_0 and using data $\mathcal{D}^n_{<i}$. Let K_∞ denote the neural tangent kernel for $\mathcal{F}$, and $\mathcal{M} = \mathrm{GP}(\mathbf{0}, K_\infty)$ the induced Gaussian Process. Then $f^i_\infty \sim P(f|\mathcal{D}^n_{<i}, \mathcal{M})$, and in the limit of infinite training time, the iterative sample-then-optimize procedure yields an unbiased estimate of $\mathcal{L}(\mathcal{D}^n|\mathcal{M})$. Letting ℓ_2 denote the scaled squared ℓ_2 regression loss and c be a constant, we obtain as a direct corollary of Theorem 4.3.3*

$$\log P(\mathcal{D}) \geq \mathbb{E}_{f^i_\infty \sim P(\cdot|\mathcal{D}_{<i})}[\log P(\mathcal{D}_i|f^i_\infty)] = \mathbb{E}\sum_{i=1}^{n} -\ell_2(\mathcal{D}_i, f^i_\infty) + c = \mathcal{L}(\mathcal{D})\ . \tag{4.9}$$

Proof. Follows immediately from the results of He et al. [2020] stating that the the limiting distribution of f^k_∞ is precisely $P(f|\mathcal{D}^n_{\leq k}, \mathcal{M})$. We therefore obtain the same result as for Theorem 4.3.3, plugging in the kernel gradient descent procedure on f for the parameter-space gradient descent procedure on θ. □

The following Lemma will be useful in order to prove Proposition B.1.1. Intuitively, this result states that in a linear regression problem in which each feature ϕ_i is 'normalized' (the dot product $\langle \phi_i, y\rangle = \langle \phi_j, y\rangle = \alpha$ for some α and all i, j) and 'independent' (i.e. $\langle \Pi_{y^\perp}\phi_i, \Pi_{y^\perp}\phi_j\rangle = 0$), then the optimal linear regression

solution assigns highest weight to the feature which obtains the least error in predicting y on its own.

Lemma B.2.1. *Let $y \in \mathbb{R}^n$, and $\Phi \in \mathbb{R}^{d \times d}$ be a design matrix such that $\Phi[:, j] = \alpha y + \epsilon_j \forall j$ for some fixed $\alpha \geq 0$, with $\epsilon \in y^{\perp}$, and $\epsilon_i^{\top} \epsilon_j = 0$ for all $i \neq j$. Let w^* be the solution to the least squares regression problem on Φ and y. Then*

$$\text{argmax}_i w_i = \text{argmin}_i \|f_i(x) - y\|^2 = \text{argmax}_i \mathcal{L}(\mathcal{M}_i) \tag{B.22}$$

Proof. We express the minimization problem as follows. We let $\phi(x) = (f_1(x), \ldots, f_k(x))$, where $f_i(x) = \alpha y + \epsilon_i$, with $\epsilon_i \perp \epsilon_j$. We denote by $\mathbb{1}$ the vector containing all ones (of length k). We observe that we can decompose the design matrix Φ into one component whose columns are parallel to y, denoted Φ_y, and one component whose columns are orthogonal to y, denoted $\Phi_{\perp}$. Let $\sigma_i^2 = \|\epsilon_i\|^2$. By assumption, $\Phi_y = \alpha y \mathbb{1}^{\top}$, and $\Phi_{\perp}^{\top} \Phi_{\perp} = \text{diag}(\sigma_1^2, \ldots, \sigma_n^2) = \Sigma$. We then observe the following decomposition of the squared error loss of a weight vector w, denoted $\ell(w)$.

$$\begin{aligned}
\ell(w) &= \|\Phi w - y\|^2 = (\Phi w - y)^{\top}(\Phi w - y) \\
&= ((\Phi_y + \Phi_{\perp})w - y)^{\top}((\Phi_y + \Phi_{\perp})w - y) \\
&= (\Phi_y w - y)^{\top}(\Phi_y w - y) + w^{\top} \Phi_{\perp}^{\top} \Phi_{\perp} w \\
&= \|y\|^2 \|1 - \alpha \mathbb{1}^{\top} w\|^2 + \sum \sigma_i^2 w_i^2
\end{aligned}$$

In particular, the loss decomposes into a term which depends on the sum of the w_i, and another term which will depend on the norm of the component of each model's predictions orthogonal to the targets y.

As this is a quadratic optimization problem, it is clear that an optimal w exists, and so $w^{\top} \mathbb{1}$ will take some finite value, say β. It is straightforward to conclude that for any fixed β, the solution to the minimization problem

$$\min_w \sum w_i^2 \sigma_i^2 : w^{\top} \mathbb{1} = \beta \tag{B.23}$$

is such that the argmax over i of w_i is equal to the index of the predictor with the minimal error. □

Proposition B.1.1. *Let $\mathcal{M}_1, \ldots, \mathcal{M}_k$ be Bayesian linear regression models with fixed noise variance σ_N^2 and Gaussian likelihoods. Let Φ be a (random) matrix of posterior prediction samples, of the form $\Phi[i,j] = \hat{y}_i^j \sim P(y_j|\mathcal{D}_{<j}, x_j, \mathcal{M}_i)$. Suppose the following two conditions on the columns of Φ are satisfied: $\mathbb{E}\langle\Phi[:,i], y\rangle = \mathbb{E}\langle\Phi[:,j], y\rangle$ for all i, j, and $\mathbb{E}\langle\Pi_{y^\perp}\phi_i, \Pi_{y^\perp}\phi_j\rangle = 0$. Let w^* denote the least-squares solution to the regression problem $\min_w \mathbb{E}_\Phi\|\Phi w - y\|^2$. Then the following holds*

$$\text{argmax}_i w_i^* = \text{argmax}_i \mathcal{L}(\mathcal{D}|\mathcal{M}_i) \qquad \forall w^* = \text{argmin}_w \mathbb{E}\|\Phi w - y\|^2 \,. \tag{B.2}$$

Proof. We begin by emphasizing the necessity of the two conditions: the first condition ensures that all of the predictions are of the same scale. This avoids situations where one feature is equal to a small multiple of the targets y (i.e. $Phi[:,i] = \beta y$ for small β) resulting in a disproportionately large weight on that model's predictions in order to normalize their scale. The second condition ensures that the models do not have complementary systematic errors, such that the optimal predictor might assign one model's predictions a higher weight in order to cancel out errors in other models' predictions. This is satisfied when we require that $\epsilon_i \perp \epsilon_j$ be orthogonal, and that ζ_i^j be sampled independently for all i and j. This assumption is crucial: in the case where there errors are linearly independent but not orthogonal, we can arrive at situations where greater weight may be assigned to a model whose error is complementary to those of other models, despite this model not being the best fit for the data.

We note that our lower bound for each model in the linear regression setting is equal to $\mathbb{E}\sum_{i=1}^N \|f_k(x_i) + \zeta_i - y_i\|^2 + c$ where c is a fixed normalizing constant. By the previous Lemma, we know that the linear regression solution w^* based on the posterior means satisfies, $\max_i w_i^* = \max_i \mathcal{L}(\mathcal{M}_i)$. It is then straightforward to

extend this result to the noisy setting.

$$\mathbb{E}[\|\Phi w - y\|^2] = \mathbb{E}[\|(\Phi_y + \Phi_\perp + \zeta)w - y\|^2] \tag{B.24}$$

$$= \mathbb{E}[((\Phi_y + \Phi_\perp + \zeta)w - y)^\top((\Phi_y + \Phi_\perp + \zeta)w - y)] \tag{B.25}$$

$$= \|\Phi_y w - y\|^2 + w^\top \Phi_\perp^\top \Phi_\perp w + \mathbb{E}[w^\top \zeta^\top \zeta w] \tag{B.26}$$

$$= (w^\top \mathbb{1} - \alpha)^2 \|y\|^2 + w^\top \Phi_\perp^\top \Phi_\perp w + \mathbb{E}[w^\top \zeta^\top \zeta w] \tag{B.27}$$

$$= (w^\top \mathbb{1} - \alpha)^2 \|y\|^2 + \sum w_i^2 (\|\Phi_\perp[:, i]\|^2 + \|\zeta_i\|^2) \tag{B.28}$$

We again note via the same reasoning as in the previous Lemma that the model with the greatest lower bound will be the one which minimizes $\|\Phi_\perp[:, i]\|^2 + \|\zeta_i\|^2$, and that the weight given to index i will be inversely proportional to this term.

It only remains to show that for each model i, the model which maximizes $\mathcal{L}(M_i)$ will also minimize $\|\Phi_\perp[:, i]\|^2 + \|\zeta_i\|^2$. This follows precisely from the Gaussian likelihood assumption. As we showed previously

$$\mathcal{L}(\mathcal{D}|\mathcal{M}_i) = \mathbb{E}[\sum \log P(y_i|\mathcal{D}_{<i})] \propto -\sum \mathbb{E}[\ell_2(y_i - \hat{y}_i] \tag{B.29}$$

$$= [\|y - \mu\|^2 + \mathbb{E}[\|\hat{y} - \mu\|^2] \tag{B.30}$$

$$= \alpha\|y\|^2 + \|\Phi_\perp[:, i]\|^2 + \mathbb{E}[\|\zeta_i\|^2] \tag{B.31}$$

and so finding the model $\mathcal{M}_i$ which maximizes $\mathcal{L}(\mathcal{D}, \mathcal{M}_i)$ is equivalent to picking the maximal index i of w^* which optimizes the expected loss of the least squares regression problem. □

B.3 Experiments

B.3.1 Experimental details: model selection using trajectory statistics

We consider 3 model selection settings in which to evaluate the practical performance of our estimators. In **prior variance selection** we evaluate a set of BLR models on a synthetic linear regression data set. Each model $\mathcal{M}_i$ has a prior distribution over the d parameters of the form $w \sim \mathcal{N}(0, \sigma_i^2 I_d)$ for some σ_i^2, and the goal is to select the optimal prior variance (in other words, the optimal regularization coefficient). We additionally evaluate an analogous initialization variance selection method on an NTK network trained on a toy regression dataset. In **frequency (lengthscale) selection** we use as input a subset of the handwritten digits dataset MNIST given by all inputs labeled with a 0 or a 1. We compute random Fourier features (RFF) of the input to obtain the features for a Bayesian linear regression model, and perform model selection over the frequency of the features (full details on this in the appendix). This is equivalent to obtaining the lengthscale of an approximate radial basis function kernel. In **feature dimension selection**, we use a synthetic dataset [Wilson and Izmailov, 2020] of the form $(\mathbf{X}, \mathbf{y})$, where $x_i = (y_i + \epsilon_1, y_i + \ldots, y_i + \epsilon_{15}, \epsilon_{16}, \ldots, \epsilon_{30})$. We then consider a set of models $\{\mathcal{M}_k\}$ with feature embeddings $\phi_k(x_i) = x_i[1, \ldots, k]$. The optimal model in this setting is the one which uses exactly the set of 'informative' features $x[1, \ldots, 15]$.

The synthetic data simulation used in this experiment is identical to that used in [Wilson and Izmailov, 2020]. Below, we provide the details.

Let k be the number of informative features and d the total number of features. We generate a datapoint $\mathcal{D}_i = \{x_i, y_i\}$ as follows:

1. Sample y_i: $y_i \sim U([0, 1])$
2. Sample k informative features: $x_{i,j} \sim N(y_i, \sigma_0) \quad \forall j \in 1, \ldots k$
3. Sample $\max(d - k, 0)$ noise features: $x_{i,k+j} \sim N(0, \sigma_1) \quad \forall j \in 1, \ldots d - k$
4. Concatenate the features: $X_i = [x_{i,1}, \ldots x_{i,d}]$

We set $\sigma_0 = \sigma_1 = 1$, $k = 15$, $n = 30$, and let d vary from 5 to n. We then run our estimators on the Bayesian linear regression problem for each feature dimension, and find that all estimators agree on the optimal number of features, k.

To compute the random fourier features used for MNIST classification, we vectorize the MNIST input images and follow the procedure outlined by Rahimi and Recht [2008] (Algorithm 1) to produce RFF features, which are then used for standard Bayesian linear regression against the binarized labels. The frequency parameter (which can also be interpreted as a transformation of the lengthscale of the RBF kernel approximated by the RFF model) is the parameter of interest for model selection.

B.3.2 Experimental details: Bayesian model comparison

Here we provide further detail of the experiment in Section 4.2.1. The goal of the experiment is to determine whether the connection between sum-over-training losses (SOTL) and model evidence observed in the linear regression setting extends to DNNs. In particular, the two sub-questions are:

1. Do models with a lower SOTL generalize better?

2. Are these models favoured by SGD?

To answer these questions, we train a linear combination of NNs. We can answer subquestion [1] by plotting the correlation between SOTL and test performance of an individual model. Further, we address subquestion [2] by considering the correlation between test loss and linear weights assigned to each model.

Below we explain the set-up of the linear combination in more detail. We train a variety of deep neural networks along with a linear 'ensemble' layer that performs a linear transformation of the concatenated logit outputs[1] of the classification models. Let $h_m(x_i)$ be logit output of model m for input x_i, $\ell(y_i, h_i)$ be the loss for point i (where h_i is a logit) and $w_{m,t}$ be the weight corresponding to model m at time step t.

[1] These are pre-softmax outputs. To obtain the predicted probability of a class, they are fed through a softmax function.

We consider two training strategies: we first train models individually using the cross-entropy loss between each model's prediction and the true label, only cross-entropy loss of the final ensemble prediction to train the linear weights. Mathematically, we update the models using the gradients

$$\frac{\partial}{\partial \theta_m} \ell(y_i, h_m(x_i)), \tag{B.32}$$

and the 'ensemble' weights using

$$\frac{\partial}{\partial w_m} \ell(y_i, \sum_m w_m h_m(x_i)). \tag{B.33}$$

We refer to this training scheme as *Parallel Training* as the models are trained in parallel. We also consider the setting in which the models are trained using the cross entropy loss from the ensemble prediction backpropagated through the linear ensemble layer, i.e. the model parameters are now updated using:

$$\frac{\partial}{\partial \theta_m} \ell(y_i, \sum_m w_m h_m(x_i)). \tag{B.34}$$

We refer to this scheme as the *Concurrent Training*.

We train a variety of different MLPs (with varying layers,and nodes) and convolutional neural networks (with varying layers, nodes and kernels) on FashionMNIST using SGD until convergence.

B.3.3 Experimental details: SGD upweights submodels that perform well

Below we provide further details of the experiment in Section 4.2.2. The goal of the experiment is to determine whether SGD upweights sub-models that fit the data better.

We train a MLP network (with units $200, 200, 10$) on FashionMMIST using SGD until convergence. After training is completed, for every class of y, we rank all nodes in the penultimate layer by the norm of their absolute weight (in the final dense layer). We group the points into submodels according to their ranking – the k nodes with the highest weights are grouped together, next the $k+1, \ldots 2k$ ranked nodes are grouped, etc. We set $k = 10$.

We determine the performance of a submodels by training a simple logistic classifier to predict the class of an input, based on the output of the submodel. To measure the performance of the classifier, we use the cross-entropy loss. To capture the equivalent notion of the AUC, we estimate the performance of the sub-models throughout training, and sum over the estimated cross-entropy losses.

Below, we show additional plots for the *parallel* and *concurrent* training schemes. The results are the same to those presented in the main text, and we observe [1] a negative correlation between test performance and ensemble weights and [2] a strong correlation between SOTL and average test cross-entropy.

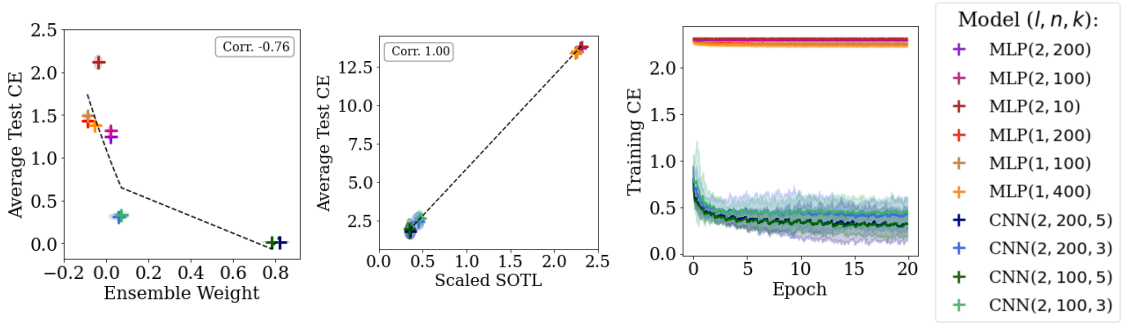


Figure B.4: Linear combinations of DNNs on FashionMNIST. Left: ensemble weights versus the test loss for parallel training; we observe a negative correlation. Middle: SOTL (standardized by the number of training samples) versus test loss for concurrent and concurrent training. We observe a strong correlation indicating that the SOTL generalizes well. Right: training curves for the different models in concurrent training schemes. All results are averaged over 10 runs, and standard deviations are shown by the shaded regions around each observation. The model parameters, given in the parentheses, are the number of layers (l), nodes per layer (n) and kernel size (k), respectively.

However, similarly to the linear setting, the difference in assigned weights is magnified in the concurrent training scheme. Here we find that in the concurrent training scheme, the ensemble focuses on training the CNNs (as can be seen from the training curve in Figure 4.5 in the main text). This is likely because CNNs are able to learn more easily, leading to larger weights earlier on.

C

Dynamics of reinforcement learning

C.1 Auxiliary results

In this section, we state and prove some additional lemmas that are useful in proving the results stated in this chapter.

Lemma C.1.1. *Let $x \in \mathbb{R}^d$, and let $(v_t)_{t\geq 0}$ be a sequence of vectors in $\mathbb{R}^d$ satisfying $v_t = f(t)x + o(f(t))$, for some function $f : [0, \infty) \to (0, \infty)$. Then $d(\langle v_t\rangle, \langle x\rangle) \to 0$ as $t \to \infty$.*

Proof. The Grassmann distance $d(\langle v_t\rangle, \langle x\rangle)$ between two one-dimensional subspaces has a particular simple form, given by

$$d(\langle v_t\rangle, \langle x\rangle) = \min\left(\arccos\left(\frac{\langle v_t, x\rangle}{\|v_t\|\|x\|}\right), \arccos\left(\frac{\langle -v_t, x\rangle}{\|v_t\|\|x\|}\right)\right) .$$

In our case, for sufficiently large t this yields

$$\begin{aligned} d(\langle v_t\rangle, \langle x\rangle) &= \arccos\left(\frac{\langle f(t)x + o(f(t)), x\rangle}{\|f(t)x + o(|f(t)|)\|\|x\|}\right) \\ &= \arccos\left(\frac{\langle x + o(1), x\rangle}{\|x + o(1)\|\|x\|}\right) \\ &\to \arccos\left(\frac{\langle x, x\rangle}{\|x\|\|x\|}\right) \\ &= 0 . \end{aligned}$$

□

Lemma C.1.2. *Let $U_1, \dots, U_{|\mathcal{X}|}$ be a basis for $\mathbb{R}^{\mathcal{X}}$, let $K < |\mathcal{X}|$, and let $(a_{ij} | i \in [K], j \in [|\mathcal{X}|])$ be real coefficients. Let $0 < \beta_1 < \cdots < \beta_{|\mathcal{X}|}$, and consider time-dependent vectors $W_1(t), \dots, W_d(t)$ defined by*

$$W_i(t) = \sum_{j=1}^{|\mathcal{X}|} a_{ij} e^{-\beta_j t} U_j \,, \quad t \geq 0 \,.$$

Then for almost all sets of coefficients $(a_{ij} | i \in [K], j \in [|\mathcal{X}|])$, we have

$$d(W_{1:K}(t), U_{1:K}) \to 0 \,.$$

Proof. Without loss of generality, we may take the vectors $U_1, \dots, U_{|\mathcal{X}|}$ to be the canonical basis vectors. Under the assumptions of the theorem, we exclude initial conditions for which the matrix A with $(i,j)^{\text{th}}$ element a_{ij} is not full rank. Note that under this condition, the matrix A_t with $(k,i)^{\text{th}}$ element $a_{ki} e^{\beta_i t}$ is also full rank for all but finitely many t. By performing row reduction operations and scaling rows, for all such t we may pass from $(W_k(t) \mid k \in [K])$ to an alternative spanning set $(\widetilde{W}_k(t) \mid k \in [K])$ of the same subspace such that $\widetilde{W}_k(t) - U_k \in \langle U_{K+1:|\mathcal{X}|} \rangle$, and $\|\widetilde{W}_k(t) - U_k\| = O(e^{-t(\beta_{K+1} - \beta_k)}) = o(1)$. We therefore obtain an orthonormal basis for this subspace of the form $U_1 + o(1), \dots, U_K + o(1)$.

We now use the singular value decomposition characterization of Grassmann distance in Definition 5.2.1. Since we have obtained an orthonormal basis for the subspace $\langle W_k(t) \mid k \in [K] \rangle$, the top-$K$ singular values of the matrix $(\sum_{k=1}^K U_k U_k^\top)(\sum_{k=1}^K (U_k + o(1))(U_k + o(1))^\top)$ determine the Grassmann distance. However, this matrix is equal to $\text{diag}(1, \dots, 1, 0, \dots, 0) + o(1)$, with K entries of 1 in the diagonal matrix. But the top-K singular values this matrix are $1 + o(1)$, and so the principal angles between the subspaces are $o(1)$, and hence the Grassmann distance between the subspaces is $o(1)$, as required. □

Lemma C.1.3. *For $M \in \mathbb{N}$, let $(r^m)_{m=1}^M$ be independent random variables drawn from some fixed mean-zero distribution in $\mathscr{P}(\mathbb{R}^{\mathcal{X} \times \mathcal{A}})$ such that the covariance between coordinates $(x,a), (y,a)$ is Σ_{xy}, independent of $a \in \mathcal{A}$. Let $(\mathbf{w}^m)_{m=1}^M$ be independent random variables taking values in $\mathbb{R}^{K \times \mathcal{A}}$, with columns drawn independently from*

$\mathcal{N}(0,(1/M)I)$. *Then* $\sum_{m=1}^{M} r^m(\mathbf{w}^m)^\top$ *converges (in distribution) to a mean-zero Gaussian distribution over* $\mathbb{R}^{\mathcal{X}\times K}$*, with independent columns, and individual columns having covariance matrix* Σ*.*

Proof. The proof simply follows by noting that $\sum_{m=1}^{M} r^m\mathbf{w}^m$ may be written $1/\sqrt{M}\sum_{m=1}^{M} r^m\varepsilon^m$, with $(\varepsilon^m)_{m=1}^{\infty}$ i.i.d. $N(0,I)$ random variables. The individual terms have the desired mean and variance, and the resulting converge in distribution now follows from the central limit theorem. □

Lemma C.1.4. *For fixed* M*, let* $(\mathbf{w}^m)_{m=1}^{M}$*,* $\mathbf{w}^m \in \mathbb{R}^d$*, be sampled i.i.d. according to* $\mathcal{N}(0,\frac{1}{M}I)$*. Then the following hold.*

$$\lim_{M\to\infty}\sum_{m=1}^{M}\mathbf{w}^m(\mathbf{w}^m)^\top = I \text{ and } \lim_{M\to\infty}\sum_{m=1}^{M}\mathbf{w}^m \overset{D}{=} \epsilon \sim \mathcal{N}(0,I) \tag{C.1}$$

Proof. We prove two results on the limit of $W = \sum_{m=1}^{M}\mathbf{w}^m(\mathbf{w}^m)^\top$ as $k\to\infty$. First

$$\lim_{M\to\infty}\sum_{m=1}^{M}\mathbf{w}^m(\mathbf{w}^m)^\top \overset{P}{=} I\,,$$

which we observe by evaluating an arbitrary diagonal and off-diagonal element of $\sum_{m=1}^{M}\mathbf{w}^m(\mathbf{w}^m)^\top$. For the diagonal terms, note that

$$\left(\sum_{m=1}^{M}\mathbf{w}^m(\mathbf{w}^m)^\top\right)[j,j] = \sum_{m=1}^{M}(\mathbf{w}_j^m)^2$$

Now observe that

$$\mathbb{E}\left[\sum_{m=1}^{M}(\mathbf{w}_j^m)^2\right] = M\frac{1}{M} = 1\,,\text{ and }\quad \mathrm{Var}\left(\sum_{m=1}^{M}(\mathbf{w}_j^m)^2\right) = M\frac{1}{M^2}\to 0$$

Similarly, for the off-diagonal terms, let $j\neq\ell$. Then we have

$$\left(\sum_{m=1}^{M}\mathbf{w}^m(\mathbf{w}^m)^\top\right)[j,\ell] = \sum_{m=1}^{M}\mathbf{w}_j^m\mathbf{w}_\ell^m\,,$$

and further

$$\mathbb{E}\left[\sum_{m=1}^{M}\mathbf{w}_j^m\mathbf{w}_\ell^m\right] = 0\,,\text{ and }\quad \mathrm{Var}\left(\sum_{m=1}^{M}\mathbf{w}_\ell^m\mathbf{w}_j^m\right) = M\frac{1}{M^2}\to 0;$$

The limit in probability is immediately implied by Chebyshev's inequality. The result on $\sum_{m=1}^{M}\mathbf{w}^m$ follows immediately from part 1 and the fact that a sum of Gaussian random variables is another Gaussian random variable whose mean and variance in this case will be a standard normal. □

C.2 Proofs

Lemma 5.3.1. *If $(V_t)_{t\geq 0}$ satisfies Equation (5.2) with initial condition V_0 at time $t = 0$, then we have*

$$V_t = \exp(-t(I - \gamma P^\pi))(V_0 - V^\pi) + V^\pi \,. \tag{5.3}$$

Proof. Equation (5.3) can be verified as a solution to Equation (5.2) by direct differentiation. Uniqueness of the solution follows since this is an autonomous initial value problem that satisfies the Lipschitz condition, and so the Picard-Lindelhöf theorem applies. □

Proposition 5.3.2. *Under Assumption 1, and $(V_t)_{t\geq 0}$ the solution to Equation (5.2), for almost every[1] initial condition V_0, we have*

$$d(\langle V_t - V^\pi \rangle, \langle U_1 \rangle) \to 0 \,.$$

Proof. By Assumption 1, P^π is diagonalisable, with eigenbasis $U_1, \ldots, U_{|\mathcal{X}|}$, with corresponding eigenvalues $\lambda_{1:|\mathcal{X}|}$ with strictly decreasing magnitudes $|\lambda_1| > \cdots > |\lambda_{|\mathcal{X}|}|$. We note then that $\exp -(t(I - \gamma P^\pi))$ is also diagonaisable under the same basis, with eigenvalues $\exp(t(\gamma\lambda_i - 1))$, for $i = 1, \ldots, |\mathcal{X}|$. We may therefore expand V_0 with respect to this eigenbasis, and write

$$V_0 - V^\pi = \sum_{i=1}^{|\mathcal{X}|} \alpha_i U_i \,,$$

for some $\alpha_{1:|\mathcal{X}|} \in \mathbb{R}^{|\mathcal{X}|}$. Now note from the differential equation (5.3), we have

$$V_t - V^\pi = \exp(-t(I - \gamma P^\pi))(V_0 - V^\pi) = \sum_{i=1}^{|\mathcal{X}|} \alpha_i \exp(t(\gamma\lambda_i - 1))U_i \,.$$

Note that as P^π is a stochastic matrix, we have $|\lambda_i| \leq 1$ for all $i = 1, \ldots, |\mathcal{X}|$, and hence $\exp(t(\gamma\lambda_i - 1)) \to 0$ for all $i = 1, \ldots, |\mathcal{X}|$. Further, $\exp(t(\gamma\lambda_i - 1)) = o(\exp(t(\gamma\lambda_1 - 1)))$ for all $i = 2, \ldots, |\mathcal{X}|$. We make the additional assumption that

[1] In the measure-theoretic sense that the set of excluded initial conditions V_0 has Lebesgue measure 0.

$\alpha_1 \neq 0$, which makes the 'almost every initial condition' assumption in the statement precise. Under this assumption, we therefore have

$$\begin{aligned} V_t - V^\pi &= \alpha_1 \exp(t(\gamma\lambda_1 - 1))U_1 + \sum_{i=2}^{|\mathcal{X}|} \alpha_i \exp(t(\gamma\lambda_i - 1))U_i \\ &= \alpha_1 \exp(t(\gamma\lambda_1 - 1))U_1 + o(\exp(t(\gamma\lambda_1 - 1)))\,. \end{aligned}$$

Then Lemma C.1.1 applies to give $d(\langle V_t - V^\pi \rangle, \langle U_1 \rangle) \to 0$, as required. □

Proposition 5.3.3. *Under Assumption 1, and $(V_t^{(k)})_{t\geq 0}$ the solution to Equation (5.2) for each $k = 1, \ldots, K$, for almost every initial condition $(V_0^{(k)})_{k=1}^K$, we have*

$$d(\langle V_t^{(k)} - V^\pi \mid k \in [K] \rangle, \langle U_{1:K} \rangle) \to 0\,.$$

Proof. Expanding $V_0^{(k)} - V^\pi$ with respect to $U_1, \ldots, U_{|\mathcal{X}|}$ for each $k = 1, \ldots, |\mathcal{X}|$, we obtain expressions of the form

$$V_0^{(k)} - V^\pi = \sum_{i=1}^{|\mathcal{X}|} a_{ki} U_i\,.$$

By the ODE solution in Lemma 5.3.1, we then have

$$V_t^{(k)} - V^\pi = \sum_{i=1}^{|\mathcal{X}|} a_{ki} e^{-t(1-\gamma\lambda_i)} U_i\,.$$

We may now apply Lemma C.1.2 to obtain the desired result. □

Lemma 5.3.4. *Let Φ_t and $\mathbf{w}_t$ parameterize a value function approximator as defined above. Then*

$$\partial_t \Phi_t = \alpha(R^\pi + \gamma P^\pi \Phi_t \mathbf{w}_t - \Phi_t \mathbf{w}_t)\mathbf{w}_t^\top\,, \tag{5.8}$$

$$\partial_t \mathbf{w}_t = \beta \Phi_t^\top (R^\pi + \gamma P^\pi \Phi_t \mathbf{w}_t - \Phi_t \mathbf{w}_t)\,. \tag{5.9}$$

Proof. This follows immediately by computing the derivatives in Equations (5.6) & (5.7), and so we omit the direct calculations. □

Theorem 5.4.1. *For $M \in \mathbb{N}$, let $(\Phi_t^M)_{t\geq 0}$ be the solution to Equation (5.10), with each $\mathbf{w}_t^m$ for $m = 1, \ldots, M$ initialized independently from $N(0, \sigma_M^2)$, and fixed throughout training ($\beta = 0$). We consider two settings: first, where the learning rate α is scaled as $\frac{1}{M}$ and $\sigma_M^2 = 1$ for all M, and second where $\sigma_M^2 = \frac{1}{M}$ and the learning rate α is equal to 1. These two settings yield the following dynamics, respectively:*

$$\lim_{M\to\infty} \partial_t \Phi_t^M \overset{P}{=} -(I - \gamma P^\pi)\Phi_t^M \quad , \text{ and} \tag{5.12}$$

$$\lim_{M\to\infty} \partial_t \Phi_t^M \overset{D}{=} -(I - \gamma P^\pi)\Phi_t^M + R^\pi \epsilon^\top \ , \ \epsilon \sim \mathcal{N}(0, I)\,. \tag{5.13}$$

The corresponding limiting trajectories for a fixed initialization $\Phi_0 \in \mathbb{R}^{\mathcal{X}\times K}$ are therefore given respectively by

$$\lim_{M\to\infty} \Phi_t^M \overset{P}{=} \exp(-t(I - \gamma P^\pi))\Phi_0 \quad , \text{ and} \tag{5.14}$$

$$\lim_{M\to\infty} \Phi_t^M \overset{D}{=} \exp(-t(I - \gamma P^\pi))(\Phi_0 - (I - \gamma P^\pi)^{-1} R^\pi \varepsilon^\top) + (I - \gamma P^\pi)^{-1} R^\pi \varepsilon^\top \ , \ \epsilon \sim \mathcal{N}(0, I)\,. \tag{5.15}$$

Proof. We write the dynamics on Φ_t^M as follows and apply the results of Lemma C.1.4. We first consider the scaled initialization setting (implicitly setting the learning rate $\alpha = 1$), where we find

$$\partial_t \Phi_t^M = (I - \gamma P^\pi)\Phi_t^M \sum_{m=1}^M \mathbf{w}^m (\mathbf{w}^m)^\top + \sum_{m=1}^M R^\pi (\mathbf{w}^m)^\top \tag{C.2}$$

$$\lim_{M\to\infty} \partial_t \Phi_t^M = (I - \gamma P^\pi)\Phi_t^M \lim_{M\to\infty} \sum_{m=1}^M \mathbf{w}^m (\mathbf{w}^m)^\top + \lim_{M\to\infty} R^\pi (\sum_{m=1}^M \mathbf{w}^m)^\top \tag{C.3}$$

$$\overset{D}{=} (I - \gamma P^\pi)\Phi_t^M I + R^\pi \epsilon^\top, \ \epsilon \sim \mathcal{N}(0, I). \tag{C.4}$$

We further observe that, for any finite interval, in the setting of zero reward we obtain *uniform* convergence of the induced trajectory Φ_t^M to the trajectory of the limiting dynamics. We first observe that for a fixed initialization, we have that the induced dynamics are linear (in the zero-reward setting, affine otherwise) function of Φ_t^M, and so

$$\partial_t \Phi_t^M = (I - \gamma P^\pi)\Phi_t^M \sum_{m=1}^M w^m (w^m)^\top = \mathcal{L}^M \Phi_t^M$$

$$\text{where } \mathcal{L}^M(A) = (I - \gamma P^\pi) A \sum_{m=1}^M w^m (w^m)^\top$$

$$\implies \Phi_t^M = \exp(t\mathcal{L}^M)\Phi_0^M \ .$$

Because the function $t \mapsto \exp(tA)$ is Lipschitz on a bounded interval for any A, this implies that for any finite interval $[0, T]$, the functions $t \mapsto \Phi_t^M$, as well as limiting solution, are L-Lipschitz for some L. Further, since the exponential is continuous,

$$\begin{aligned}\lim_{M\to\infty} \Phi_t^M &= \lim_{M\to\infty} \exp(t\mathcal{L}^M)\Phi_0^M = \exp(t \lim_{M\to\infty} \mathcal{L}^M)\Phi_0 \\ &= \exp(-t(I - \gamma P^\pi))\Phi_0 = \Phi_t^\infty \,.\end{aligned}$$

Therefore, the functions $t \mapsto \Phi_t^M$ are L-Lipschitz and converge to the limit Φ_t^∞ on the interval $[0, T]$, which implies that they converge uniformly.

To evaluate the scaled learning rate setting, we observe that we now have

$$\partial_t \Phi_t^M = \frac{1}{M}(I - \gamma P^\pi)\Phi_t^M \sum_{m=1}^M \mathbf{w}^m(\mathbf{w}^m)^\top + \sum_{m=1}^M R^\pi(\mathbf{w}^m)^\top \tag{C.5}$$

$$\lim_{M\to\infty} \partial_t \Phi_t^M = (I - \gamma P^\pi)\Phi_t^M \lim_{M\to\infty} \frac{1}{M} \sum_{m=1}^M \mathbf{w}^m(\mathbf{w}^m)^\top + \lim_{M\to\infty} \frac{1}{M} R^\pi(\sum_{m=1}^M \mathbf{w}^m)^\top \tag{C.6}$$

$$= (I - \gamma P^\pi)\Phi_t^M I. \tag{C.7}$$

$$\implies \lim_{M\to\infty} \Phi_t^M = \exp(-t(I - \gamma P^\pi))\Phi_0 \,, \tag{C.8}$$

almost surely. The principal difference between this and the scaled initialization setting is that here we divide the $R^\pi \mathbf{w}^\top$ term by $\frac{1}{M}$, whereas the scaled initialization is equivalent to scaling by $\frac{1}{\sqrt{M}}$. Therefore the scaled learning rate limit can be computed by the law of large numbers and converges in probability to its mean (zero), whereas under the scaled initialization it converges via the central limit theorem to a Gaussian distribution. □

Corollary 5.4.2. *Under the feature flow (5.10) with $\mathbf{w}_t^m$ fixed at initialization for each $i = 1, \ldots, M$, under either the scaled initialization or scaled learning rate assumption of Theorem 5.4.1, and under Assumption 1, for almost all initializations Φ_0, we have when $R^\pi = 0$*

$$d(\langle \Phi_t \rangle, \langle U_{1:K} \rangle) \to 0 \,, \quad \text{as } t \to \infty.$$

Proof. As described in the proof of Theorem 5.4.1, we have $\Phi_t = \exp(-t(I - \gamma P^\pi))(\Phi_0 - \Phi_\infty) + \Phi_\infty$. Under Assumption 1, we may now apply an analogous argument as in Proposition 5.3.3 to the columns of $\Phi_t - \Phi_\infty$, and apply Lemma C.1.2 to obtain the desired result. □

Theorem 5.4.3. *For fixed $M \in \mathbb{N}$, let the random rewards $(r^m)_{m=1}^M$ and weights $(\mathbf{w}^m)_{m=1}^M$ be as defined above, let $\alpha = 1$, and consider the representation dynamics in Equation (5.19), with weights fixed throughout training ($\beta = 0$). Let Σ denote the covariance matrix of the random cumulant distribution. Then*

$$\lim_{M\to\infty} \sum_{m=1}^{M} r^m (\mathbf{w}^m)^\top \overset{D}{=} Z_\Sigma \sim \mathcal{N}(0, \Sigma), \text{ and}$$
$$\lim_{M\to\infty} \Phi_t^M \overset{D}{=} \exp(-t(I - \gamma P^\pi))(\Phi_0 - (I - \gamma P^\pi)^{-1} Z_\Sigma) + (I - \gamma P^\pi)^{-1} Z_\Sigma \,.$$

As the columns of Z_Σ are mean-zero, uncorrelated, with covariance matrix Σ, the limiting distribution of each column of $\Phi_\infty = \lim_{t\to\infty} \lim_{M\to\infty} \Phi_t^M$ has covariance $\Psi\Sigma\Psi^\top$, where Ψ is the resolvent $(I - \gamma P^\pi)^{-1}$.

Proof. We recall from Theorem 5.4.1 that the limiting dynamics follow the distribution

$$\lim_{t\to\infty} \lim_{M\to\infty} \Phi_t \overset{D}{=} \lim_{t\to\infty} \exp(-t(I - \gamma P^\pi))(\Phi_0 - (I - \gamma P^\pi)^{-1} Z_\Sigma) + (I - \gamma P^\pi)^{-1} Z_\Sigma \tag{C.9}$$
$$\overset{D}{=} (I - \gamma P^\pi)^{-1} Z_\Sigma \tag{C.10}$$

for which we can straightforwardly apply known properties of Gaussian distributions: namely, that the distribution of a linear transformation A of a Gaussian random variable with parameters μ, Σ is also Gaussian with mean $A\mu$ and covariance $A\Sigma A^\top$. Letting $A = (I - \gamma P^\pi)$ therefore gives the desired result. □

Corollary 5.4.4. *Under the feature flow (5.19) with $\mathbf{w}_t^m$ fixed at initialization for each $i = 1, \ldots, M$ and Assumption 1, for almost all initializations Φ_0, we have when $R^\pi = 0$*

$$d(\lim_{M\to\infty} \langle \Phi_t^M - \Phi_\infty \rangle, \langle U_{1:K} \rangle) \to 0 \,, \quad \text{as } t \to \infty.$$

Proof. As described in the proof of Theorem 5.4.3, we have $\Phi_t = \exp(-t(I - \gamma P^\pi))(\Phi_0 - (I - \gamma P^\pi)^{-1} Z_\Sigma) + (I - \gamma P^\pi)^{-1} Z_\Sigma$. Under Assumption 1, we may now apply an analogous argument as in Proposition 5.3.3 to the columns of $\Phi_t - (I - \gamma P^\pi)^{-1} Z_\Sigma$, and apply Lemma C.1.2 to obtain the desired result. □

C.3 Additional results from Table 5.1

We begin this section by noting the following property of systems following linear dynamics.

Lemma C.3.1. *Let $\Phi_t \in \mathbb{R}^{\mathcal{X} \times M}$ follow the dynamics $\partial_t \Phi_t \overset{D}{=} A\Phi_t + B$, where A is a linear operator for which all eigenvalues have negative real part, and B is a vector. Then*

$$\lim_{t \to \infty} \Phi_t = -A^{-1} B \,. \tag{C.11}$$

Further, if A is diagonalisable, with all eigenvalues of different magnitudes,

$$\lim_{t \to \infty} d(\langle \Phi_t - \Phi_\infty \rangle, \langle U_{1:K}(A) \rangle) = 0 \,, \tag{C.12}$$

where $U_i(A)$ is the eigenvector of A corresponding to the eigenvalue with i^{th} largest magnitude.

Proof. We observe that the dynamics $\partial_t \Phi_t = A\Phi_t$ induce the trajectory

$$\Phi_t = \exp(tA)\Phi_0 + (I - \exp(tA))(-A^{-1}B) \,, \tag{C.13}$$

with limit $\Phi_\infty = -A^{-1}B$. When A is diagonalizable, we can therefore straightforwardly apply the results of Lemma C.1.2 to get that the limiting subspace will be characterized by the top k eigenvectors of A. In the settings we are interested in, $A = -(I - \gamma P^\pi)$ for some π and some γ, and so the principal eigenvectors of A will be the principal eigenvectors of P^π. □

The following two theorems characterize the learning dynamics under the past policies and multiple timescale auxiliary tasks listed in Table 5.1. With these characterizations, it becomes straightforward to deduce Φ_∞ and the limiting subspace error as a direct consequence of the previous lemma.

Theorem C.3.2. *Let $\pi_1, \ldots, \pi_L$ be a fixed set of policies. Given fixed M and L, we define the indexing function $i_m = \lceil \frac{L}{m} \rceil$ for $m \in [1, M]$. Let Φ_t^M follow the dynamics*

$$\partial_t \Phi_t^M = \sum_{m=1}^{M} -((I - \gamma P^{\pi_{i_m}})\Phi_t^M \mathbf{w}_t^m + R^{\pi_{i_m}})(\mathbf{w}_t^m)^\top \tag{C.14}$$

Then Φ_t^M satisfies the following dynamics and trajectory in the limit as $M \to \infty$, where $\bar{\pi} = \sum_{i=1}^{L} \pi_i$ and $\epsilon_i \in \mathbb{R}^d$ is an isotropic Gaussian with variance $\frac{1}{L}$. Note that we cannot naively average the rewards without changing the variance of the induced distribution unless $R^{\pi_i} = R^{\pi_j}$ for all i, j.

$$\lim_{M\to\infty} \partial_t \Phi_t^M \overset{D}{=} -(I - \gamma P^{\bar{\pi}})\Phi_t + \sum_{i=1}^{L} R^{\pi_i} \epsilon_i \tag{C.15}$$

$$\lim_{M\to\infty} \Phi_t^M \overset{D}{=} \exp(-t(I - \gamma P^{\bar{\pi}}))(\Phi_0 - \Phi_\infty) + (I - \gamma P^{\bar{\pi}})^{-1}\left(\sum_{i=1}^{L} R^{\pi_i} \epsilon_i^\top\right) \tag{C.16}$$

Proof. The result on the trajectories follows immediately from the result on the dynamics, so it suffices to prove convergence of the dynamics. We approach this problem by decomposing the dynamics of Φ_t^M as follows.

$$\partial_t \Phi_t^M = \sum_{m=1}^{M} -(I - \gamma P^{\pi_{i_m}})\Phi_t^M \mathbf{w}_t^m (\mathbf{w}_t^m)^\top - \sum_{m=1}^{M} R^{\pi_{i_m}} (\mathbf{w}_t^m)^\top . \tag{C.17}$$

We first consider the random variables in the term which includes the rewards R^π. For this, we can directly apply the results from the previous theorems to the random variables $\epsilon_j = \sum_{m:i_m=j} w^m$, whose limiting variance is easily computed to be

$$\lim_{M\to\infty} \mathrm{Var}\left(\sum_{m:i_m=j} \mathbf{w}^m\right) = \lim_{M\to\infty} \sum_{\lfloor \frac{j}{n} M \rfloor}^{\lfloor \frac{j+1}{n} M \rfloor} \frac{1}{M} I = \frac{1}{L} I . \tag{C.18}$$

For the term which depends on Φ_t, we see

$$\sum_{m=1}^{M} -(I - \gamma P^{\pi_{i_m}})\Phi_t^M \mathbf{w}_t^m (\mathbf{w}_t^m)^\top = \sum_{i=1}^{L} \sum_{m:i_m=i}^{M} -(I - \gamma P^{\pi_{i_m}})\Phi_t^M \mathbf{w}_t^m (\mathbf{w}_t^m)^\top \tag{C.19}$$

$$= \sum_{i=1}^{L} -(I - \gamma P^{\pi_i})\Phi_t^M \sum_{m:i_m=i}^{M} \mathbf{w}_t^m (\mathbf{w}_t^m)^\top . \tag{C.20}$$

Since L is finite and fixed, $\sum_{m:i_m=i}^{M} \mathbf{w}_t^m (\mathbf{w}_t^m)^\top$ converges to $\frac{1}{L} I$

$$\xrightarrow[M\to\infty]{} \sum_{i=1}^{L} -(I - \gamma P^{\pi_i})\Phi_t^M \frac{1}{L} I \tag{C.21}$$

$$= -(I - \gamma \frac{1}{L} \sum_{i=1}^{L} P^{\pi_i})\Phi_t^M \tag{C.22}$$

$$= -(I - \gamma P^{\bar{\pi}})\Phi_t^M \,. \tag{C.23}$$

And so the limiting distribution becomes

$$\lim_{M\to\infty} \partial_t \Phi_t^M = -(I - \gamma P^{\bar{\pi}})\Phi_t^M - \left(\sum_{i=1}^{L} R^{\pi_i} \epsilon_i \right). \tag{C.24}$$

□

Corollary C.3.3. *The above result can be readily adapted to the setting in which each head predicts a randomly selected (deterministic) policy in MDPs with finite state and action spaces. Let $L = |\mathcal{A}|^{|\mathcal{X}|}$, $\{\pi_1, \ldots, \pi_L\}$ be an enumeration of $\mathcal{A}^{\mathcal{X}}$, and i_m denote the index of the policy randomly assigned to head m; then the above result still holds, and $\bar{\pi}$ is the uniform policy.*

Theorem C.3.4. *We consider the task of predicting the value functions of a fixed policy under multiple discount rates $\gamma_1, \ldots, \gamma_L$. For fixed M, L, let i_m denote the indexing function defined in Theorem C.3.2 Let Φ_t follow the dynamics*

$$\partial_t \Phi_t^M = \sum_{m=1}^{M} -((I - \gamma_{i_m} P^\pi)\Phi_t^M \mathbf{w}_t^m + R^\pi)(\mathbf{w}_t^m)^\top \,. \tag{C.25}$$

Then the limiting dynamics as $M \to \infty$ of Φ_t^M are as follows, where $\bar{\gamma} = \sum \frac{1}{L}\gamma_i$

$$\lim_{M\to\infty} \partial_t \Phi_t^M \overset{D}{=} -(I - \bar{\gamma} P^\pi)\Phi_t + R^\pi \epsilon^\top \tag{C.26}$$

and

$$\lim_{M\to\infty} \Phi_t^M \overset{D}{=} \exp(-t(I - \gamma P^{\bar{\pi}}))(\Phi_0 - \Phi_\infty) + (I - \gamma P^\pi)^{-1} R^\pi \epsilon^\top) \,. \tag{C.27}$$

Proof. We follow a similar derivation as for Theorem C.3.2 in deriving the component of the dynamics which depends on Φ_t^M. The result of Theorem 5.4.1 immediately

applies to the $\sum R^\pi(\mathbf{w}_t^m)^\top$ term:

$$\sum_{m=1}^{M} -(I-\gamma_{i_m}P^\pi)\Phi_t^M\mathbf{w}_t^m(\mathbf{w}_t^m)^\top = \sum_{i=1}^{L}\sum_{m:i_m=i}^{M} -(I-\gamma_i P^\pi)\Phi_t^M\mathbf{w}_t^m(\mathbf{w}_t^m)^\top \tag{C.28}$$

$$= \sum_{i=1}^{L} -(I-\gamma_i P^\pi)\Phi_t^M \sum_{m:i_m=i}^{M}\mathbf{w}_t^m(\mathbf{w}_t^m)^\top . \tag{C.29}$$

Since L is finite and fixed, $\sum_{m:i_m=i}^{M}\mathbf{w}_t^m(\mathbf{w}_t^m)^\top$ converges to $\frac{1}{L}I$ as before:

$$\xrightarrow[M\to\infty]{} \sum_{i=1}^{L} -(I-\gamma_i P^\pi)\Phi_t^M\frac{1}{L}I \tag{C.30}$$

$$= -(I-\sum_{i=1}^{L}\frac{\gamma_i}{L}P^\pi)\Phi_t^M \tag{C.31}$$

$$= -(I-\bar{\gamma}P^\pi)\Phi_t^M . \tag{C.32}$$

□

C.4 Beyond diagonalisability assumptions

In this section, we briefly describe extensions of the results of Chapter 5 in scenarios where Assumption 1 does not hold. There are two main cases we consider: (i) those in which P^π is still diagonalisable, but does not have all eigenvalues with distinct magnitudes; and (ii) those in which P^π is not diagonalisable.

In the former case, we do not have the different convergence rates of coefficients of different eigenvectors as in the proof of Proposition 5.3.3. By similar arguments we can still deduce convergence of V_t to the span of the eigenspaces with highest magnitude eigenvalues, but we can no longer deduce convergence to individual eigenspaces if there are several other eigenvalues with the same magnitude as the eigenvalue concerned. Note also that this includes the case where the matrix P^π is complex- but not real-diagonalisable, since in such case non-real eigenvalues must come in conjugate pairs (which are necessarily of the same absolute value).

In the latter case, we no longer have an eigenbasis for $\mathbb{R}^\mathcal{X}$ based on P^π. However, we can consider the Jordan normal decomposition, and may still recover analogous results to those in Chapter 5, where convergence is now to the subspaces generated by *Jordan blocks* with high absolute value eigenvalues. See Parr et al. [2008] for further commentary on Jordan normal decompositions in feature analysis.

C.5 Extensions beyond one-step temporal difference learning

Our analysis in Chapter 5 has focused on the case of learning dynamics under one-step temporal difference learning. This choice is largely because one-step temporal difference learning is such a popular algorithm, not because the results do not hold more generally. In this section, we describe the elements of analogous results for n-step learning and TD(λ) for interested readers. We focus on the case of value function dynamics, and believe extensions of the representation dynamics analysis in Chapter 5 along these lines will be interesting directions for future work.

C.5.1 Temporal difference learning with n-step returns

In the case of n-step returns, the dynamics on the value function $(V_t)_{t\geq 0}$ are given by

$$\partial_t V_t(x) = \mathbb{E}_\pi \left[\sum_{k=0}^{n-1} \gamma^k R_k + \gamma^n V_t(X_n) \middle| X_0 = x \right] - V_t(x) \,.$$

In full vector notation, we have

$$\partial_t V_t = -(I - \gamma^n (P^\pi)^n) V_t + \left[\sum_{k=0}^{n-1} (\gamma P^\pi)^k \right] R^\pi \,.$$

The solution to this differential equation is

$$V_t = \exp(-t(I - (\gamma P^\pi)^n))(V_0 - V^\pi) + V^\pi \,.$$

This bears a close relationship with the result obtained for 1-step temporal difference learning in Chapter 5. As expected, we obtain the same limit point. Further, under Assumption 1, $(P^\pi)^n$ has the same eigenvectors as P^π, and so results analogous to Propositions 5.3.2 & 5.3.3 hold for n-step temporal difference learning too under these conditions.

C.5.2 Temporal difference learning with λ-returns

In the case of temporal difference learning with λ-returns (for $\lambda \in [0, 1)$), the dynamics on the value function $(V_t)_{t\geq 0}$ are given by

$$\partial_t V_t(x) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} (\lambda\gamma)^k (P^\pi)^k (R^\pi + \gamma P^\pi V_t(X_{k+1}) - V_t(X_k)) \middle| X_0 = x \right] - V_t(x) \,.$$

In full vector notation, we have

$$\partial_t V_t = \sum_{k=0}^{\infty} (\lambda\gamma)^k (P^\pi)^k (R^\pi + \gamma P^\pi V_t - V_t)$$

The solution to this differential equation is

$$V_t = \exp\left(t\left((1-\lambda)\sum_{k=1}^{\infty}\lambda^{k-1}\gamma^k(P^\pi)^k - I\right)\right)(V_0 - V^\pi) + V^\pi\,.$$

As with n-step temporal difference learning, this bears a close relationship with the result obtained for 1-step temporal difference learning in Chapter 5. As expected, we obtain the same limit point. Further, under Assumption 1, each $(P^\pi)^k$ has the same eigenvectors as P^π, and so results analogous to Propositions 5.3.2 & 5.3.3 hold for n-step temporal difference learning too under these conditions.

C.6 Bayes-optimality of RSBFs

We can develop the discussion of RSBFs beyond their properties as a matrix decomposition described in Section C.7 to observe that the RSBFs characterize the Bayes-optimal features for predicting an unknown value function given an isotropic Gaussian prior distribution on the reward, and further characterize a Bayesian posterior over value functions given by conditioning on the known dynamics of the MDP. We will denote by $V_K(\Psi)$ the top K eigenvectors of the matrix $\Psi\Psi^\top$, i.e. the top K left singular vectors of Ψ.

Corollary C.6.1. *Under an isotropic Gaussian prior on reward function $r \in \mathbb{R}^{\mathcal{X}}$, the subspace $V_K(\Psi)$ corresponds to the optimal subspace with respect to the following regression problem.*

$$\min_{\Phi\in\mathbb{R}^{\mathcal{X}\times K}} \mathbb{E}_{r\sim\mathcal{N}(0,I)}\left[\|\Pi_{\Phi^\perp}(I-\gamma P^\pi)^{-1}r\|^2\right], \tag{C.33}$$

where $\Pi_{\Phi^\perp}$ denotes orthogonal projection onto the orthogonal complement of Φ.

Proof. Let S denote some subspace $S \subset V$.

$$\mathbb{E}[\|\Pi_s \Psi r\|^2] = \mathbb{E}[r^\top \Psi^\top \Pi_s^\top \Pi_s \Psi r] \tag{C.34}$$

We note that for any real symmetric matrix A we can rewrite $A = \sum \alpha_i v_i v_i^\top$.

$$\mathbb{E}[r^\top \Psi^\top \Pi_s^\top \Pi_S \Psi r] = \mathbb{E}[r^\top (\sum \alpha_i v_i v_i^\top) r] = \mathbb{E}[\sum \alpha_i (r^\top v_i)(v_i^\top r)] \tag{C.35}$$

$$= \mathbb{E}[\sum \alpha_i v_i^\top r r^\top v_i] = \sum \alpha_i v_i^\top \mathbb{E}[r r^\top] v_i \tag{C.36}$$

$$= \sum \alpha_i v_i^\top v_i = \mathrm{Tr}(\Psi^\top \Pi_S^\top \Pi_S \Psi) = \mathrm{Tr}(\Psi^\top \Pi_S \Psi) \tag{C.37}$$

Finally, we can re-express the minimization problem as follows

$$\mathrm{argmin}_{S:\mathrm{Dim}(S)=k} \mathrm{Tr}(\Psi^\top (\Pi_{S^\perp}) \Psi) = \mathrm{argmax}_{S:\mathrm{Dim}(S)=k} \mathrm{Tr}(\Psi^\top \Pi_S \Psi) \tag{C.38}$$

Now, because the subspace spanned by the top k left-singular vectors $\{u_1, \ldots, u_k\}$ of Ψ is known to be the maximizer of the above equation, we finally obtain

$$= \langle u_1, \ldots, u_k \rangle = V_K(\Psi) \, . \tag{C.39}$$

□

Corollary C.6.2. *The limiting distribution of Φ_t^M under the random cumulant auxiliary task described in Theorem 5.4.3 is equivalent to the Bayesian posterior over value functions obtained by conditioning on the dynamics P^π, and given a prior distribution on the reward function equal to $\mathcal{N}(0, \Sigma)$.*

Proof. Each column of Z_Σ is sampled from an isotropic Gaussian distribution, and therefore each feature $\phi_i \overset{D}{=} (I - \gamma P^\pi)\epsilon_i$. It therefore suffices to show that under a suitable prior distribution, the distribution of ϕ_i is equal to a Bayesian posterior. For this, it suffices to show that such a posterior can be obtained by conditioning on the transition dynamics P^π, and looking at the induced pushforward measure on the reward distribution. Noting that $(I - \gamma P^\pi)$ is invertible, we then obtain the following prior over V^π, assuming an isotropic Gaussian prior on $p_r(r)$ and any arbitrary distribution over potential transition dynamics $p_\pi(P^\pi)$ which covers $\mathbb{R}^{|S| \times d}$.

$$P(V^\pi) = \int_{(r, P^\pi)} \mathbb{1}[(I - \gamma P^\pi)^{-1} r = V^\pi] dp_r(r) dp_\pi(P^\pi) \tag{C.40}$$

We observe that the random variable V^π has conditional distribution $P(V^\pi|P^\pi) = P((I - \gamma P^\pi)^{-1}r)$, whose density is proportional to $p_r((I - \gamma P^\pi)V)$ by the change of variables formula.

$$P(V^\pi|P^\pi) = cp_r(r = (I - \gamma P^\pi)V^\pi) \tag{C.41}$$

Because our prior over r is equal to the initialization distribution of ϵ_i, we obtain

$$= cp_{\text{init}}(\epsilon_i = (I - \gamma P^\pi)V^\pi) \tag{C.42}$$

which is precisely the limiting distribution p_∞ of ϕ_i (again applying the change of variables formula).

$$= p_\infty(\phi_i = (I - \gamma P^\pi)^{-1}\epsilon_i = V^\pi) \tag{C.43}$$

So we see that the limiting distribution of ϕ_i is equal to the prior over value functions conditioned on the transition dynamics. □

C.7 Further discussion of features and operator decompositions

Proto-value functions (PVFs), were first defined by Mahadevan and Maggioni [2007] as the eigenvectors of the *incidence matrix* induced by the environment transition matrix P. In the ensuing years, the term PVF has been used to refer to a number of related but not necessarily equivalent concepts. To clarify our use of the term and the relationship of our decompositions of the resolvent and transition matrices of an MDP, we provide a brief discussion here; a summary is provided in Table C.1.

We will use A to refer to the adjacency matrix of the unweighted, undirected graph induced by the matrix P (i.e. $A[i, j]$ is 1 if there exists some action with nonzero probability of taking the agent from state i to state j or from state j to state i, and 0 otherwise). L_G will refer to the graph Laplacian based on this matrix A.

We can additionally consider the Laplacian of the weighted, directed graph defined by P^π; we will refer to this matrix as L_{P^π}, in reference to its dependence on the probability of transitioning. T denotes the matrix defined by a collection

of sampled transitions indexed by t, with entries $T_{it} = -1$ if the transition t leaves i and $+1$ if it enters state i.

Our first observation is that eigendecomposition and SVD are equivalent for symmetric matrices because any real symmetric matrix has an orthogonal eigenbasis; this means that performing either decomposition yields the same eigenvectors and easily related eigenvalues. Our second observation is that when P^π is *not* symmetric, its singular value decomposition and eigendecomposition may diverge; further, the relationship between the SVD of the resolvent matrix $\Psi = (I - \gamma P^\pi)^{-1}$ and of P^π is no longer straightforward, despite the eigenspaces of the two matrices being analogous. This means that analysis of the singular value decomposition of P^π does not immediately imply any results about the resolvent matrix.

Matrix	SVD	Eigendecomposition (ED)
L_G	PVFs [Mahadevan and Maggioni, 2007]	Equivalent to SVD
T	sometimes $\equiv$ ED(L_G) [Machado et al., 2017]	not discussed
L_{P^π}	$\neq$ ED(L_{P^π})	Stachenfeld et al. [2014]
$(I - \gamma P^\pi)^{-1}$	RSBFs	$\equiv L_{P^\pi}$
P^π	Behzadian and Petrik [2018]	$\equiv L_{P^\pi}$

Table C.1: Summary of decompositions of various matrices associated with MDP transition operators, and associated features.

Finally, we note that applying a uniform random walk policy may not be sufficient to guarantee that P^π will be symmetric, and that in general it will not be possible to obtain a policy which will symmetrize the transition matrix. For example: when G is a connected, non-regular graph (as is the case in many environments such as chains), there must be a node v of degree d adjacent to a node v' of degree $d' \neq d$. A random walk policy will assign $p(v, v') = \frac{1}{d}$, while $p(v', v)$ will receive probability $\frac{1}{d'}$; thus, P^π will not be symmetric. Fortunately, this is not a barrier to spectral analysis; the eigenvectors and eigenvalues of P^π will still be real, as their transition matrix will be *similar* to a symmetric matrix. We defer to Machado et al. [2017] for a more detailed discussion of this relationship.

C.8 Learning dynamics for ensemble prediction

We provide some visualizations of the induced behaviour on features as a result of training an ensemble with multiple heads and zero reward, replicating the analysis of Section 5.2.1, to highlight how the eigendecomposition of P^π affects the learned representations. We run our evaluations on the Four-Rooms Gridworld by initializing $\Phi \in \mathbb{R}^{105\times 10}$ (i.e. $|\mathcal{X}| = 105$ and the number of features $d = 10$) and simulating the ODE defined in Equation 5.10 for time $t = 100$ with transition matrix P^π defined by the uniform random policy on this Gridworld. In some cases, the features converged to zero quickly and so we show a final $t < 100$ to highlight the behaviour of the representation before it reaches zero.

We consider three variables which we permit to vary: the initialization scheme of features, in one case sampled from an isotropic Gaussian `rand` or from a randomly initialized 2-layer MLP `nn`); whether the weight matrix is fixed at initialization `fix` or permitted to follow the flow defined by Equation 5.7 `train`; and finally the number of 'heads', `M`=1, 20, and 200.

In Figure C.1, we plot the output of an arbitrary head $\mathbf{w}^m$ of the ensemble. In Figure C.2 we visualize the value of a single feature (i.e. a single column of Φ).

We observe, as predicted, that for fixed heads in the overparameterized regime, the features (and the value functions they induce) converge to smooth eigenfunctions. We do not see meaningful convergence of the features trained in conjunction with a single weight vector. In contrast, the value functions and features trained in conjunction with ensembles with more heads than the feature dimension consistently resemble the eigenfunctions of P^π. When $(\mathbf{w}^m)$ are held fixed, we see convergence to smooth eigenfunctions as predicted by our theory; when $(\mathbf{w}^m)$ are permitted to vary according to the flow in Equation 5.11, we see convergence to the most eigenfunction corresponding to the most negative eigenfunction of P^π.

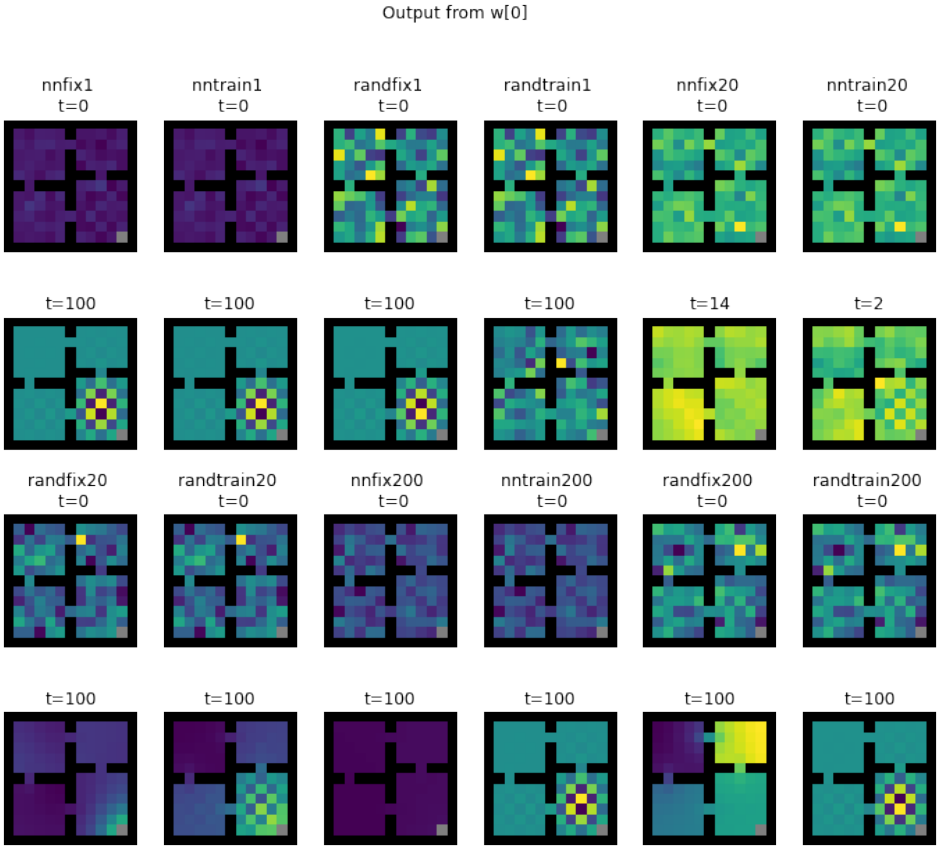


Figure C.1: Value functions learned by the ensemble head at index 0 for different training regimes. Plot titles of form (feature initialization scheme, train/fix weight matrix, number of heads in ensemble). Observe that the representation learned with fixed weights tends to converge to smoother eigenfunctions than those learned with weights that are also allowed to train. f

C.9 Experimental details

C.9.1 Experimental details for Figure 5.3

In our evaluations of the evolution of single feature vectors, we compute the continuous-time feature evolution defined in Equation (5.6), using P^π defined by a random walk on a simple Four-Rooms Gridworld with no reward. We use a randomly initialized representation $\Phi \in \mathbb{R}^{|\mathcal{X}|\times 10}$, and use a single column of this matrix in our feature visualization (we observed similar behaviour in each feature). To compute trajectories, we use the SciPy ODE solver `solve_ivp` [Virtanen et al., 2020].

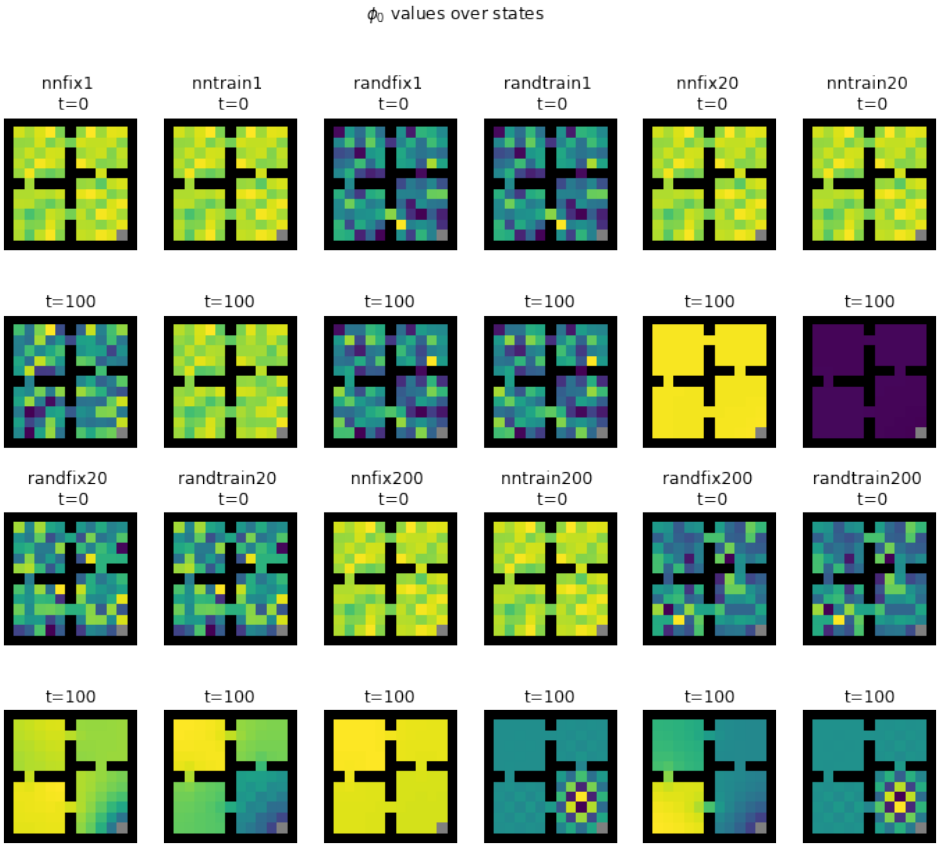


Figure C.2: Values of ensemble feature at index 0 for different training regimes. Plot titles of form (feature initialization scheme, train/fix weight matrix, number of heads in ensemble). Observe that the representation learned with fixed weights tends to converge to smoother eigenfunctions than those learned with weights that are also allowed to train.

C.9.2 Experimental details for Section 5.5.1

Here, we provide details of the environment used in producing Figure 5.5.1. The environment is a 30-state chain, with two actions, `left` and `right`, which move the agent one state to the left or right, respectively. When the agent cannot move further left or right (due to being at an end state of the chain), the result of the corresponding action keeps the agent in the same state. There is additionally environment stochasticity of 0.01, meaning that with this probability, a uniformly random action is executed instead. This stochasticity ensures that P^π satisfies the conditions of Assumption 1. Taking the action left in the left-most state incurs a reward of +2, and taking the action `right` in the right-most state incurs a reward of +1; all other rewards are zero.

C.9.3 Experimental details for Section 5.5.2

We modify a base Double DQN agent [Van Hasselt et al., 2016] and evaluate on the ALE without sticky actions [Bellemare et al., 2013]. Our agents are implemented in Jax [Bradbury et al., 2018], and are based on the DQN Zoo [Quan and Ostrovski, 2020]. Unless otherwise mentioned, all hyperparameters are as for the default Double DQN agent, with the exception of the epsilon parameter in the evaluation policy, which is set to 0.001 in all agents, and the optimizer, which for agents using auxiliary tasks CV, REM and Ensemble is Adam with epsilon $0.1/32^2$, and a lightly tuned learning rate; see below for further details.

Experimental results shown in bar plots, such as Figures 5.6 and C.4, report a "relative score" which is the per-game score normalized by the maximum average score achieved by any agent or configuration. The same, per-game, normalization values are used for all such figures.

Auxiliary task details. In this section, we describe the implementations of all auxiliary tasks considered in the main text.

- *QR-DQN.* The implementation and hyperparameters match QR-DQN-1 in Dabney et al. [2018].

- *DDQN+RC.* We use a many-head DQN network which is identical to the standard neural network used for DQN, except that the output dimension is $(M+1) \times |\mathcal{A}|$ instead of $|\mathcal{A}|$, where M is the number of auxiliary heads. Random cumulants are generated using a separate neural network with the same architecture as a standard DQN, but with output dimension equal to the number of auxiliary heads. The width of the Huber loss for each auxiliary head is equal to the number of auxiliary tasks. Let $\phi(x) \in \mathbb{R}^M$ be the output of the cumulant network given input observation x, with M the number of auxiliary heads. Then the cumulant for auxiliary head m, at time step t, is given by $c_t = s \times (\phi(x_{t+1}) - \phi(x_t))$, where $s \in \mathbb{R}$ is a scaling factor. We performed a small hyperparameter sweep over scaling factors in $\{1, 10, 100, 500\}$, finding $s = 100$ to provide the best performance and use this value for all reported

experiments. Note that this auxiliary task and the details are nearly identical to the *CumulantValues* auxiliary task of Dabney et al. [2021], except that we do not pass the values through a tanh non-linearity as this did not appear to have any impact in practice. We performed a hyperparameter sweep over learning rates and gradient norm clipping for this agent, considering learning rates $\{0.00025, 0.0001, 0.00005\}$ and gradient clipping in $\{10, 40\}$. We found that a learning rate of 0.00005 and gradient norm clipping of 40 to work best and use these values for all experiments.

- *DDQN+REM.* We use a many-head variant of Double DQN, with heads trained according to the REM loss of Agarwal et al. [2020]. For the agent's policy, an argmax over a uniform average of the heads is used. We swept over learning rates of 0.0001 and 0.00005, generally finding 0.00005 to perform best.

- *DDQN+Ensemble.* As for the REM auxiliary task, we use a many-head variant of Double DQN. Each head is trained using its own double DQN loss, and the resulting losses are averaged. For the agent's policy, an argmax over a uniform average of the heads is used. We swept over learning rates of 0.0001 and 0.00005, generally finding 0.00005 to perform best.

Modified dense-reward games. We modified four Atari games (Pong, MsPacman, Seaquest, and Q*bert) to obtain sparse, harder versions of these games to test the performance of random cumulants and other auxiliary tasks. The details of these games are given below. In each case a low-valued, commonly encountered reward is 'censored', which means that during training the agent observes a reward of 0 instead of the targeted reward. When evaluated, and thus for all empirical results reported, the standard uncensored rewards are reported.

- *Sparse Pong.* All negative rewards are censored (i.e. set to 0 before being fed to the agent), so the agent receives a reward of +1 for scoring against the opponent, but no reward when it concedes a point to the opponent. As

0, 1, and −1 are the only rewards in Pong, this modification makes Pong significantly harder. The agent can no longer learn to 'avoid losing points', but can only improve by learning to score points directly.

- *Sparse MsPacman.* All rewards less than or equal to 10 are censored. This corresponds to rewards for the numerous small pellets that MsPacman eats, but not the larger pellets or ghosts. Each level ends when all of the small pellets are consumed, thus, by hiding these from the agent we may have significantly changed the primary incentive for the agent to advance the game.

- *Sparse Seaquest.* All rewards less than or equal to 20 are censored. This corresponds to the rewards for shooting the sharks underwater, but not the rewards for picking up divers or surfacing. Additionally, even the rewards for sharks increase beyond this level, and thus become visible, once the agent has surfaced and collected enough divers.

- *Sparse Q*bert.* All rewards less than or equal to 25 are censored. These are the rewards for flipping the colour of a tile, which is the primary source of reward and the mechanism for advancing to the next level of the game. Once all tiles are flipped, the agent will go to the next level. However, the agent can still observe rewards for going to the next level and for dispatching the enemies.

As described in the main text, we found that the sparse versions of MsPacman, Seaquest, and Q*bert were too difficult for any agent we tested to achieve a reasonable level of performance. In Figure C.3, we display the performance of several auxiliary tasks on these games, noting that the performance achieved is extremely low in comparison to the agents trained on the standard versions of these games (see Section 5.5.2).

Hyperparameter sweeps. In Figure C.4 we vary the weight of the auxiliary loss for the random cumulants agents, with the aim of understanding how this hyperparameters affect each method's performance. Next, in Figures C.5 and C.6 we

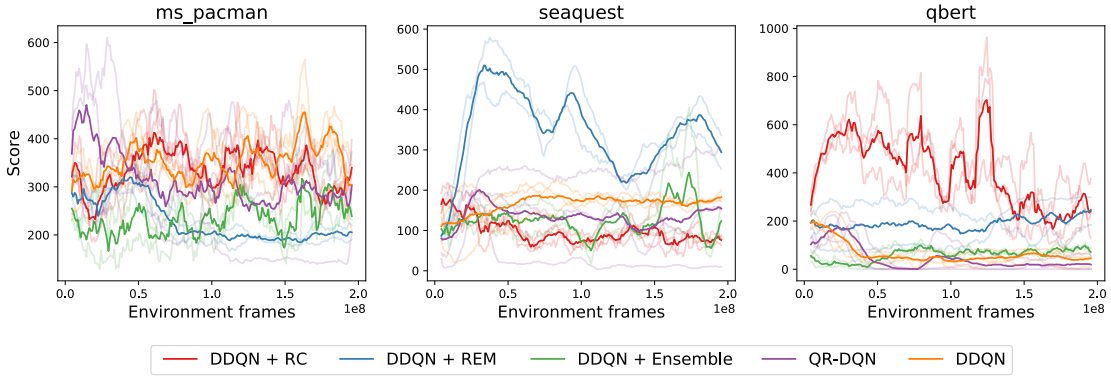


Figure C.3: Learning curves on sparsified MsPacman (left), sparsified Seaquest (centre), and sparisifed Q*bert (right).

present the results of a hyperparameter sweep for Ensemble and REM respectively. For these two, since there is no separate auxiliary loss as in RC, we vary number of heads and the learning rate. Results presented in the main text use the best settings for each algorithm found from these sweeps.

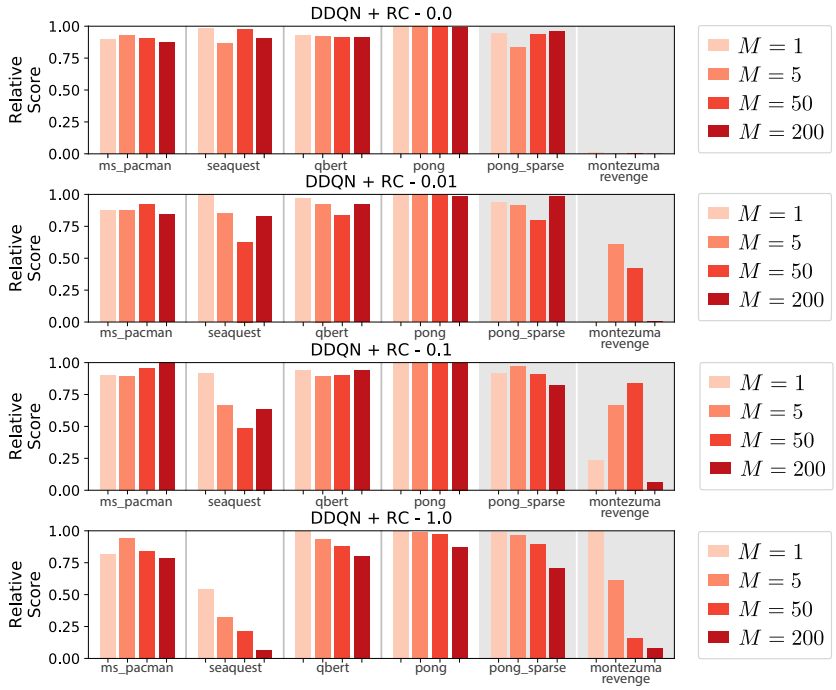


Figure C.4: Results of hyper-parameter sweep for Random Cumulant (RC) method, where each row is for a different value of multiplicative scale applied to the auxiliary losses and each bar corresponds to the number of auxiliary heads (M). Note that the first row of results corresponds to initializing a network with the auxiliary heads, but setting the weight to zero, effectively disabling the auxiliary task.

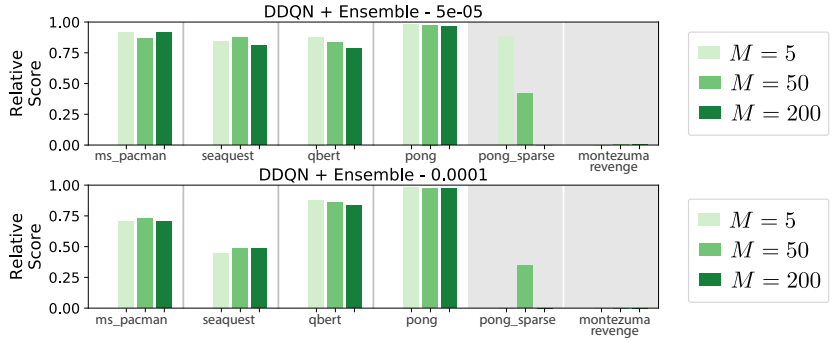


Figure C.5: Results of hyper-parameter sweep for the Ensemble method, where each row is for a different learning rate and each bar corresponds to the number of auxiliary heads (M).

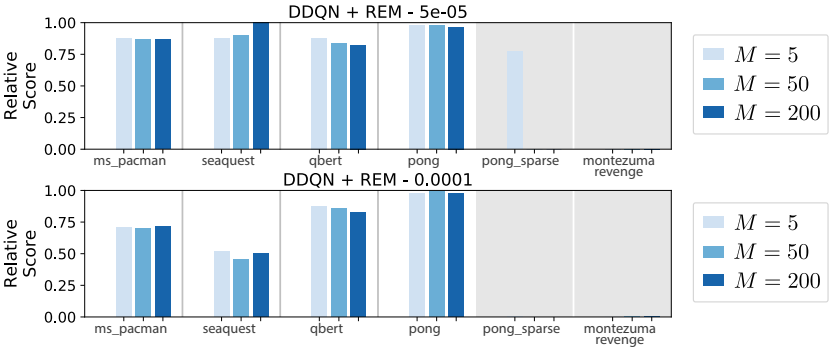


Figure C.6: Results of hyper-parameter sweep for the REM method, where each row is for a different learning rate and each bar corresponds to the number of auxiliary heads (M).

D
Capacity loss

D.1 Proofs

D.1.1 Estimator consistency

We here show that our estimator of the agent's feature rank is consistent. First recall

$$\left(\frac{1}{\sqrt{n}}\Phi_n\right)^\top \left(\frac{1}{\sqrt{n}}\Phi_n\right) = \frac{1}{n}\sum_{i=1}^{n}\phi(x_i)\phi(x_i)^\top . \tag{D.1}$$

The following property of the expected value holds

$$\mathbb{E}_{x\sim P}[\phi(x)\phi(x)^\top] = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\phi(x_i)\phi(x_i)^\top\right] . \tag{D.2}$$

It is then straightforward to apply the strong law of large numbers. To be explicit, we consider an element of $M = \mathbb{E}[\phi\phi^\top]$, M_{ij}.

$$\mathbb{E}[(\phi(x)\phi(x)^\top)_{ij}] = M_{ij} = \mathbb{E}[\phi_i(x)\phi_j(x)] \implies \sum_{k=1}^{n}\frac{1}{n}\phi_i(x_k)\phi_j(x_k) \xrightarrow{a.s.} M_{ij} . \tag{D.3}$$

Since we have convergence for any M_{ij}, we get convergence of the resulting matrix to M. Because the singular values of Φ are the eigenvalues of M and the eigenvalues are continuous functions of that matrix, the eigenvalues of M_n converge to those of M almost surely. Then for almost all values of ϵ, the threshold estimator $N(\lambda_1, \ldots, \lambda_k; \epsilon) = |\{\lambda_i > \epsilon\}|$ will converge to $N(\text{spec}(M); \epsilon)$. Specifically, the estimator will be convergent for all values of ϵ which are not eigenvalues of M itself.

D.1.2 Feature collapse case study: quantile regression

We apply similar analysis to that of Chapter 5 to better understand the effect of sparse-reward environments on representation collapse. To do so, we return to the setting where Φ_t are non-parametric features and w_t a linear function approximator which jointly parameterize a value function $V_t = \langle \Phi_t(x), w_t \rangle$. We recall the dynamics

$$\partial_t \Phi_t = \alpha(\gamma P^\pi - I)\Phi_t(w_t w_t^\top) + R^\pi w_t^\top \tag{D.4}$$

and

$$\partial_t w_t = \beta \Phi_t^\top [(\gamma P^\pi - I)\Phi_t w_t + R^\pi], \tag{D.5}$$

where $P^\pi \in \mathbb{R}^{\mathcal{X}\times\mathcal{X}}$ is the matrix of state-transition probabilities under π, and $R^\pi \in \mathbb{R}^{\mathcal{X}}$ is the vector of expected rewards.

In value-based deep RL, we model a Q-function which takes as input an observation $\mathbf{x}$ and outputs a vector $Q(s, a_i)_{i=1}^{n_a} \in \mathbb{R}^{n_a}$. This resembles the setting of *ensemble prediction*, whose dynamics we also recall here.

$$\partial_t \Phi_t^M = \alpha \sum_{m=1}^{M} (R^\pi + \gamma P^\pi \Phi_t^M w_t^m - \Phi_t^M w_t^m)(w_t^m)^\top, \tag{D.6}$$

$$\partial_t w_t^m = \beta (\Phi_t^M)^\top (R^\pi + \gamma P^\pi \Phi_t^M w_t^m - \Phi_t w_t^m). \tag{D.7}$$

The ensemble learning regime also bears similarity to the quantile regression DQN (QR-DQN) objective [Dabney et al., 2018], which learns to fit a set of quantiles to the distribution taken by the return when treated as a random variable. Each quantile $\tau_i \in (0, 1)$ is trained using a quantile regression loss, of the form

$$\mathcal{L}_{\mathrm{QR}}^\tau(\theta) := \mathbb{E}_{\hat{Z}\sim Z}[\rho_\tau(\hat{Z} - \theta)] \quad \text{where } \rho_\tau(u) = u(\tau - \delta_{u<0}),\ \forall u \in \mathbb{R}.$$

This loss is known to yield unbiased gradients, however it has the undesirable property that the gradients of $\mathcal{L}_{\mathrm{QR}}^\tau$ remain constant even as $\theta \to \tau$, which can result in pathological convergence properties. The QR-DQN objective addresses this by implementing a Huber loss [Huber, 1964], whose gradients tend to zero at its minimum.

$$\mathcal{L}_\kappa(u) = \begin{cases} \frac{1}{2}u^2 & \text{if } |u| \le \kappa \\ \kappa(|u| - \frac{1}{2}\kappa) & \text{otherwise.} \end{cases} \tag{D.8}$$

For a sufficiently large number of quantiles and under similar conditions as in Theorem 5.4.1 the dynamics induced by this objective bear marked resemblance to those of the ensemble prediction setting. In particular, when no reward is observed in the environment, we again obtain that the features will converge to the zero vector. The setting of this result is distinct from that of deep neural network representation dynamics, as neural networks use discrete optimization steps, finite learning rates, and typically do not leverage linear ensembles. However, we emphasize two crucial observations that suggest the intuition developed in this setting may be relevant: first, in sparse reward environments the representation will be pushed to zero along dimensions spanned by the linear weights used to compute outputs. Once sufficiently many independent weight vectors are being used to make predictions, this effectively forces *every* dimension of the representation to fit the zero vector output. We would therefore expect representation collapse to be particularly pronounced in the QR-DQN agents trained on sparse-reward environments, as in this setting we obtain many independently initialized heads all identically trying to fit the zero target.

Theorem D.1.1. *Let let* $\boldsymbol{\tau} = (\tau_i)_{i=1}^M$ *be a set of quantile functions* $\tau_i : \mathcal{X} \to \mathbb{R}$ *with corresponding estimators* $(\hat{\tau}^i)_{i=1}^M$ *of the form* $\hat{\boldsymbol{\tau}} = \hat{\tau}^i(\mathbf{x}) = \langle \phi(\mathbf{x}), \mathbf{w}^i \rangle$ *for* $\mathbf{w}^i \in \mathbb{R}^d$, $\phi : \mathcal{X} \to \mathbb{R}^d$. *Let* π *be some policy in an MDP* $\mathcal{M}$, *with* $R^\pi = \mathbf{0}$, *and* P^π *be the corresponding transition function. Let* Φ_t^M, $\mathbf{w}_t^i$ *follow the dynamics*

$$\partial_t \Phi_t = \beta^M \nabla_\Phi \mathcal{L}_\kappa(\hat{\boldsymbol{\tau}}) \tag{D.9}$$

$$\partial_t \mathbf{w}_t = \alpha^M \nabla_{\mathbf{w}} \mathcal{L}_\kappa(\hat{\boldsymbol{\tau}}) \; . \tag{D.10}$$

Then if $\alpha^M = 0$ *or* $\alpha^M = O(\frac{1}{M})$, *letting* $\Phi_\infty = \lim_{t\to\infty} \Phi_t$, *we obtain*

$$\lim_{M\to\infty} \Phi_\infty \to \mathbf{0}. \tag{D.11}$$

Proof. We break the derivation of this result into two cases. We first observe that, when $\alpha = 0$ and $\beta = 1$, the QR-DQN update reduces $\|V_t\|_\infty$ by either κ if $\|(\gamma P^\pi - I)V_t\|_\infty > \kappa$ or $\gamma\|V_t\|_\infty$ if $\|(\gamma P^\pi - I)V_t\|_\infty < \kappa$. In the latter case, the QR-DQN objective is identical to the TD updates studied in Chapter 5, and we

obtain convergence of Φ to zero in the case of infinitely many heads, and in the case of $M = d$ heads with orthogonal initialization aligned with the features.

In the case where $M \to \infty$, when w^i are initialized to scale with $\frac{1}{\sqrt{M}}$, we obtain straightforwardly that $\|\Phi_0 w^i\| \xrightarrow{P} 0$. Thus, the limiting probability that any error falls outside the κ cutoff of the Huber loss will be zero, and the dynamics on Φ will be identical to the TD dynamics studied previously, allowing us to apply the result of Theorem 5.4.1.

In the case where $M = d$ and the w^i are initialized such that $\langle \phi^i, w^j \rangle = \delta_{i,j}\phi^i$, we need only show that the dynamics followed by quantile regression are sufficiently well-behaved to push $\|V_t\|_\infty < \kappa$ for some finite t.

We note that in this case we can express the dynamics as follows, and for the rest of this discussion assume without loss of generality that $\phi_t^i(x) > 0 \forall x \in \mathcal{X}$.

$$\partial_t \Phi_t = -\sum_{i=1}^{M} \min((I - \gamma P^\pi)\Phi_t w^i, \kappa)(w^i)^\top \tag{D.12}$$

$$= -\sum_{i=1}^{M} \min((I - \gamma P^\pi)\phi_t^i w^i, \kappa)(w^i)^\top \tag{D.13}$$

$$\implies \partial_t \phi_t^k = --\min((I - \gamma P^\pi)\phi_t^k w^k, \kappa)(w^k)^\top \tag{D.14}$$

$$= -\min((I - \gamma P^\pi)\phi_t^k, \kappa) \tag{D.15}$$

Let ϕ_t^i be a feature vector at some time t. Let $x_{\max}$ maximize $\{|\phi_t^i(x)| : x \in \mathcal{X}\}$. Then we know that the time-derivative on $\partial_t \phi_t^i(x_{\max})$ must have the opposite sign to $\phi_t^i(x_{\max})$, as it takes the following form.

$$\partial_t \phi_t^i(x_{\max}) = -\min((I - \gamma P^\pi)\phi_t^i(x_{\max}), \kappa) \tag{D.16}$$

$$\text{sign}(\partial_t \phi_t^i(x_{\max}) = -\text{sign}(\phi_t^i(x_{\max}) \tag{D.17}$$

Now, because this applies to the maximal value of ϕ_t^i over all states, we get that the function $\tilde{\phi}^i(t) = \max_{x \in \mathcal{X}} \phi_t^i(x)$ is decreasing. The rate of this decrease is either constant, in which case it is equal to κ, or we get the standard exponential convergence described in 7.4. □

Key insight.
In the presence of many prediction targets, such as those induced by a quantile regression objective, sparse-reward environments will induce low dimensional feature representations in deep RL agents.

D.2 Sequential supervised learning

D.2.1 Details: target-fitting capacity on non-stationary MNIST

In addition to our evaluations in the Atari domain, we also consider a variant of the MNIST dataset in which the labels change over the course of training.

- **Inputs and Labels:** We use 1000 randomly sampled input digits from the MNIST dataset and assign either binary or random targets.
- **Distribution Shift:** We divide training into $N = 30$ or $N = 10$ iterations depending on the structure of the target function. In each iteration, a target function is randomly sampled, and the network's parameters obtained at the end of the previous iteration are used the initial values for a new optimization run. We use the Adam [Kingma and Ba, 2015] optimizer with learning rate `1e-3`, and train to minimize the mean squared error between the network outputs and the targets for either 3000 or 5000 steps depending on the nature of the target function.
- **Architecture:** we use a standard fully-connected architecture with ReLU activations, and vary the width and depth of the network. The parameters at the start of the procedure are initialized following the defaults in the Jax Haiku library.

We note that the dataset sizes, training budgets, and network sizes in the following experiments are all relatively small. This was chosen to enable short training times and decrease the computational budget necessary too replicate the experiments. The particular experiment parameters were selected to be the fastest and cheapest settings in which we could observe the capacity loss phenomenon,

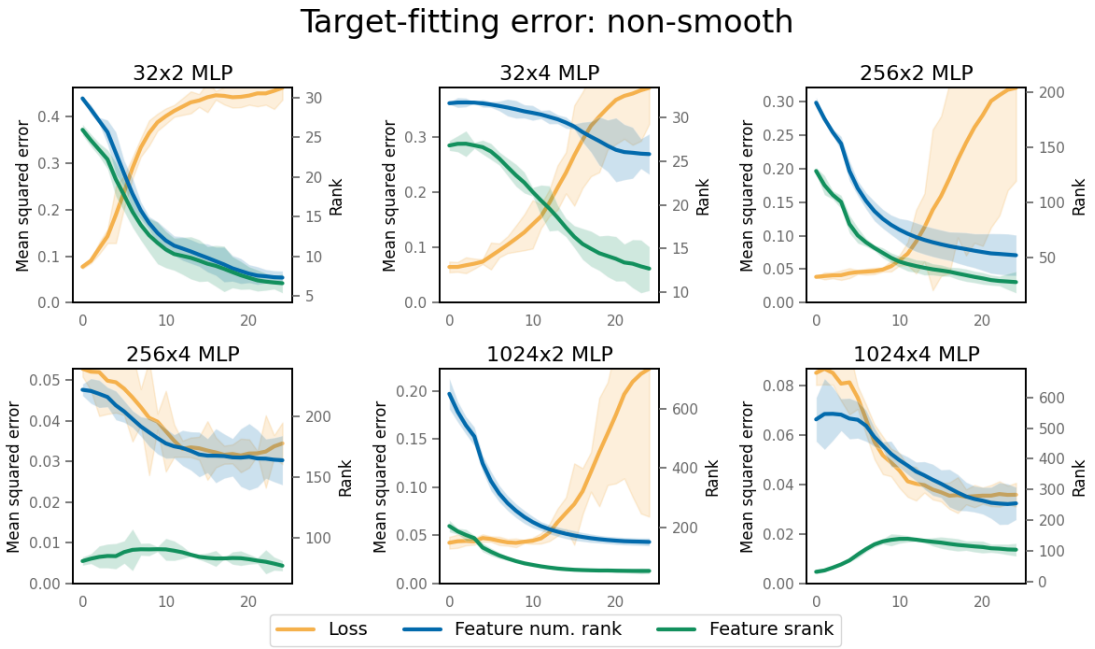


Figure D.1: Mean squared error at the end of training on each iteration of the **hash-MNIST** task. Target-fitting error increases over time in smaller networks, but increasing the depth or width of the network slows down capacity loss, enabling positive transfer in the largest networks we studied.

while still being nontrivial tasks. In general, we found that capacity loss is easiest to measure in a 'sweet spot' where the task for a given architecture is simple enough for a freshly-initialized network to attain low loss, but complex enough that the network cannot trivially solve the task. In the findings of the following section, we see how some of the larger architectures do not exhibit capacity loss on 'easier' target functions, but do on more challenging ones that exhibit less structure. This suggests that replicating these results in larger networks will be achievable, but will require re-tuning the task difficulty to the larger network's capacity.

D.2.2 Additional evaluations

We expand on the MNIST target-fitting task shown in the main text by considering how network size and target function structure influences capacity loss.

- **Random-MNIST** (smooth) this task uses the images from the MNIST dataset as inputs. The goal is to perform regression on the outputs of a randomly initialized, fixed neural network. We use a small network for this

task, consisting of two width-30 fully connected hidden layers with ReLU activations which feed into a final linear layer which outputs a scalar. Because the network outputs are small, we scale them by 10 so that it is not possible to get a low loss by simply predicting the network's bias term. This task, while randomly generated, has some structure: neural networks tend to map similar inputs to similar outputs, and so the inductive bias of the targets will match that of the function approximator we train on them.

- **Hash-MNIST** (non-smooth) uses the same neural network architecture as the previous task to generate targets, however rather than using the scaled network output as the target, we multiply the output by 1e3 and feed it into a sine function. The resulting targets no longer have the structure induced by the neural network. This task amounts to memorizing a set of labels for the input points.

- **Threshold-MNIST** (sparse) replaces the label of an image with a binary indicator variable indicating whether the label is smaller than some threshold. To construct a sequence of tasks, we set the threshold at iteration i to be equal to i. This means that at the first iteration, the labels are of the form $(x, 0)$ for all inputs x. At the second iteration, they are of the form $(x, \delta(y < 1))$, where y is the digit in the image x, and so on.

We consider MLP networks of varying widths and depths, noting that the network architecture used to generate the random targets is fixed and independent of the approximating architecture. We are interested in evaluating whether factors such as target function difficulty, network parameterization, and number of target functions previously fit influence the network's ability to fit future target functions. Our results are shown in Figure D.1, D.2, and D.3. We visualize srank and feature rank of the features output at the network's penultimate layer, in addition to the loss obtained at the end of each iteration.



Figure D.2: Mean squared error after 2e3 training steps on the **random-MNIST** task. Target-fitting error increases over time in under-parameterized networks, but increasing the depth or width of the network slows down capacity loss, enabling positive transfer in the largest network we studied.



Figure D.3: Mean squared error after 2e3 training steps on the **threshold-MNIST** task. Target-fitting error increases over time in under-parameterized networks, but increasing the depth or width of the network slows down capacity loss, enabling positive transfer in the largest network we studied.

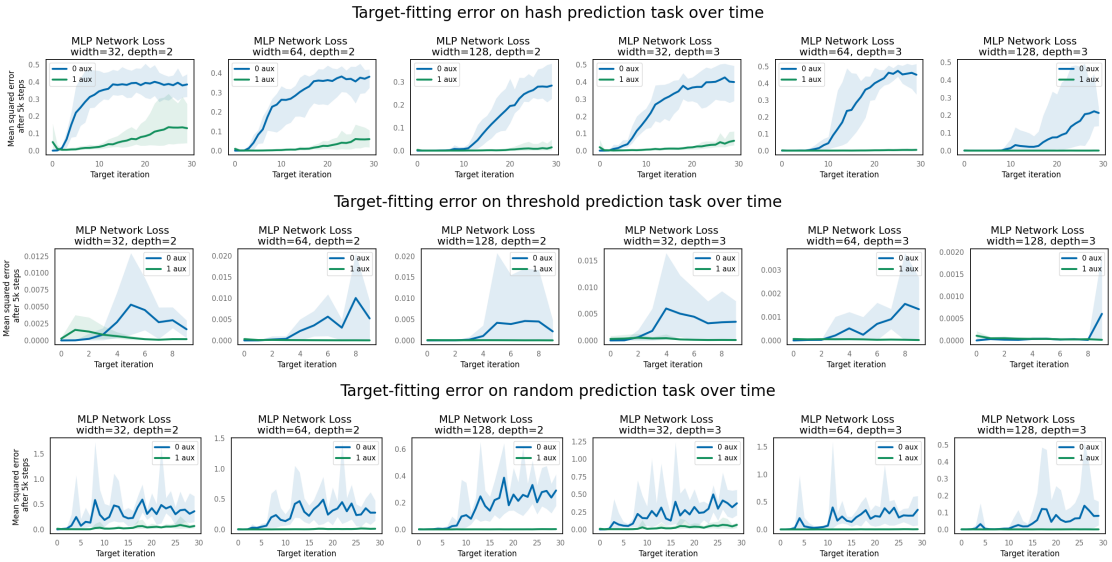


Figure D.4: Effect of adding InFeR to the regression objective in a random reward prediction problem on the non-stationary MNIST environment studied previously. We see that the InFeR objective produces networks that can consistently outperform those trained with a standard regression objective, exhibiting minimal capacity loss in comparison to the same network architecture trained on the same sequence of targets.

D.2.3 Effect of InFeR on target-fitting capacity in MNIST

In addition to our study of the Atari suite, we also study the effect of InFeR on the non-stationary MNIST reward prediction task with a fully-connected architecture; see Figure D.4. We find that it significantly mitigates the decline in target-fitting capacity demonstrated in Figure 6.1.

D.3 Atari evaluations

We now present full evaluations of many of the quantities described in the corresponding chapter, along with a study of the sensitivity of InFeR to its hyperparameters. We use the same training procedure for all of the figures in this section, loading agent parameters from checkpoints to compute the quantities shown.

D.3.1 Hyperparameter sensitivity of InFeR in deep reinforcement learning agents

We report results of hyperparameter sweeps over the salient hyperparameters relating to InFeR, so as to assess the robustness of the method. For both the DDQN and

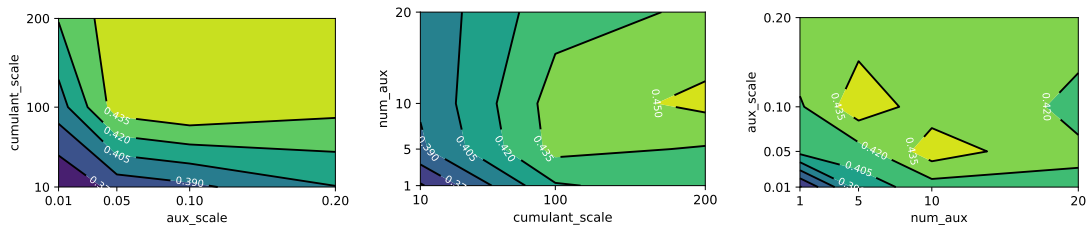


Figure D.5: Hyperparameter sweeps for the DDQN+InFeR agent. Each contour plot shows average capped human-normalized score at the end of training marginalized over all hyperparameters not shown on its axes.

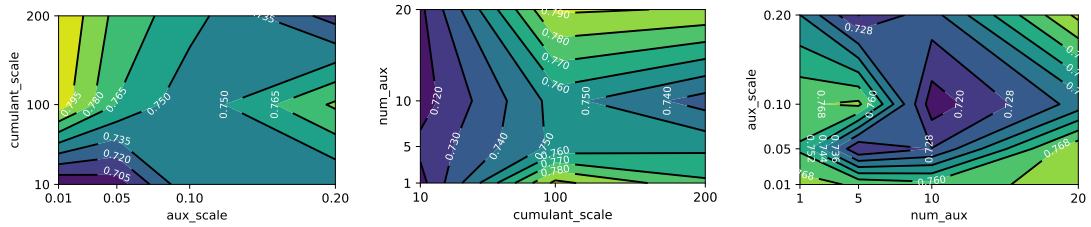


Figure D.6: Hyperparameter sweeps for the Rainbow+InFeR agent. Each contour plot shows average capped human-normalized score at the end of training marginalized over all hyperparameters not shown on its axes.

Rainbow agents augmented with InFeR, we sweep over the number of auxiliary predictions (1, 5, 10, 20), the cumulant scale used in the predictions (10, 100, 200), and the scale of the auxiliary loss (0.01, 0.05, 0.1, 0.2). We consider the capped human-normalized return across four games (Montezuma's Revenge, Hero, James Bond, and MsPacman), and run each hyperparameter configuration with 3 seeds. Results are shown in Figure D.5 for the DDQN agent; we compare performance as each pair of hyperparameters varies (averaging across the other hyperparameter, games, and seeds, and the last five evaluation runs of each agent). Corresponding results for Rainbow are given in Figure D.6.

- **Agent:** We train a Rainbow agent [Hessel et al., 2018] with the same architecture and hyperparameters as are described in the open-source implementation made available by Quan and Ostrovski [2020]. We additionally add InFeR, as described in Section 6.3, with 10 heads, gradient weight 0.1 and scale 100.

- **Training:** We follow the training procedure found in the Rainbow implementation mentioned above. We train for 200 million frames, with 500K evaluation

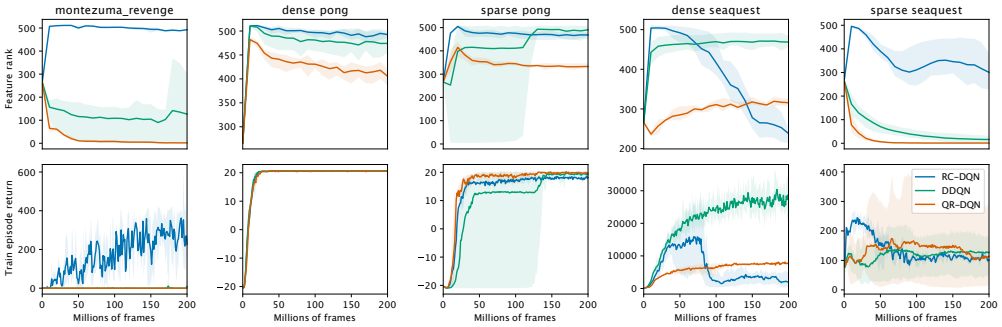


Figure D.7: Feature rank and performance of RL agents on demonstrative Atari environments.

frames interspersed every 1M training frames. We save the agent parameters and replay buffer every 10M frames to estimate feature dimension and target-fitting capacity.

D.3.2 Feature rank

We first extend the results shown in Figure 6.3 to two additional games: Seaquest, and a sparsified version of Pong in which the agent does not receive negative rewards when the opponent scores. In these settings, we stored agent checkpoints once every 10M frames in each 200M frame trajectory, and used 5000 sampled inputs from the agent's replay buffer to estimate the feature rank, using the cutoff $\epsilon = 0.01$. Results are shown in Figure D.7.

We further evaluate the evolution of feature rank in agents trained on all 57 games in the arcade learning environment in Figure D.8. We find that the decline in dimension after the first checkpoint at 10M frames shown across the different agents in the selected games also occurs more generally in Rainbow agents across most environments in the Atari benchmark. We also show that in most cases adding InFeR mitigates this phenomenon. Our observations here do not show a uniform decrease in feature rank or a uniformly beneficial effect of InFeR. The waters become particularly muddied in settings where neither the Rainbow nor Rainbow+InFeR agent consistently make learning progress such as in tennis, solaris, and private eye. It is outside the scope of this work to identify precisely why the agents do not make

learning progress in these settings, but it does not appear to be due to the type of representation collapse that can be effectively prevented by InFeR.

Procedure. We compute the feature rank by sampling $n = 50000$ transitions from the replay buffer and take the set of origin states as the input set. We then compute a $n \times d$ matrix whose row i is given by the output of the penultimate layer of the neural network given input S_i. We then take the singular value decomposition of this matrix and count the number of singular values greater than 0.01 to get an estimate of the dimension of the network's representation layer.

In most games, we see a decline in feature rank after the first checkpoint at 10M frames. Strikingly, this decline in dimension holds even in the online RL setting where the agent's improving policy presumably leads it to observe a more diverse set of states over time, which under a fixed representation would tend to increase the numerical rank of the feature matrix. This indicates that even in the face of increasing state diversity, agents' representations face strong pressure towards degeneracy. It is worth noting, however, that the agents in dense-reward games do tend to see their feature rank increase significantly early in training; this is presumably due to the network initially learning to disentangle the visually similar states that yield different bootstrap targets.

D.3.3 Target-fitting capacity early in training

In this section we examine the target-fitting capacity of neural networks trained with DQN, QR-DQN, and Rainbow over the course of 50 million environment frames on five games in the Atari benchmark (amidar, montezuma's revenge, pong, bowling, and hero). Every 1 million training frames we save a checkpoint of the neural network weights and replay buffer. For each checkpoint, we generate a random target network by initializing network weights with a new random seed. We then train the checkpoint network to predict the output of this random target network for 10000 mini-batch updates (batch size of 32) under a mean squared error loss, for states sampled from the first $100,000$ frames in the checkpoint's

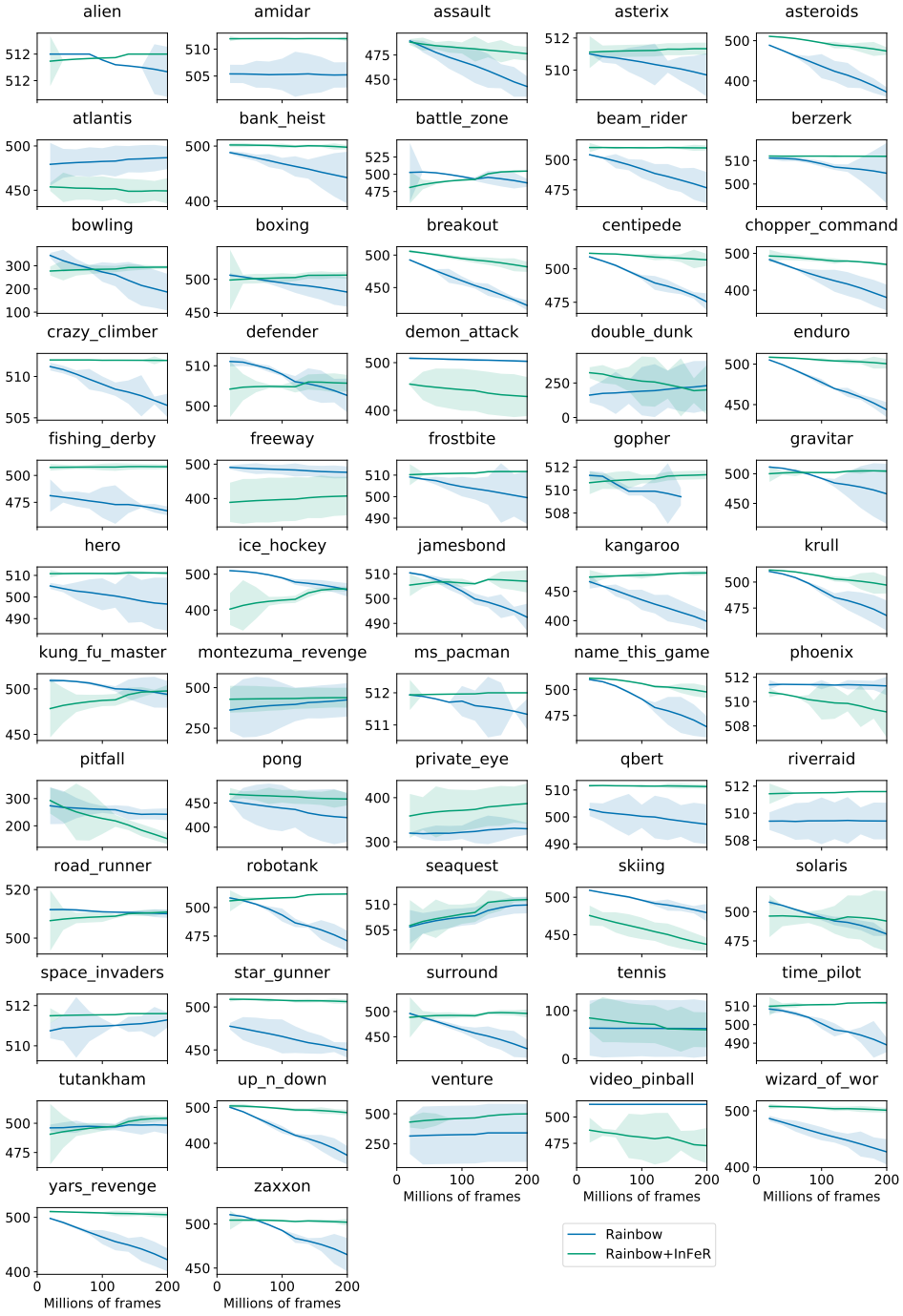


Figure D.8: feature rank of agent representations over the course of training on all 57 games in the Atari benchmark. We compare Rainbow against Rainbow+InFeR.

replay buffer. Furthermore, we repeat this for 10 seeds used to initialize the random target network weights.

The results of this experiment are shown in Figure D.9 (in orange), where the solid lines show means and shaded regions indicate standard deviations over all seeds (both agent seeds (5) and target fitting seeds (10), for a total of 50 trials). We also show srank and feature rank of the features output at the network's penultimate layer for each of the checkpointed networks used for target fitting. These are computed using the network features generated from 1000 states sampled randomly from that checkpoint's replay buffer. For feature rank, averages and standard deviations are only over the 5 agent seeds.

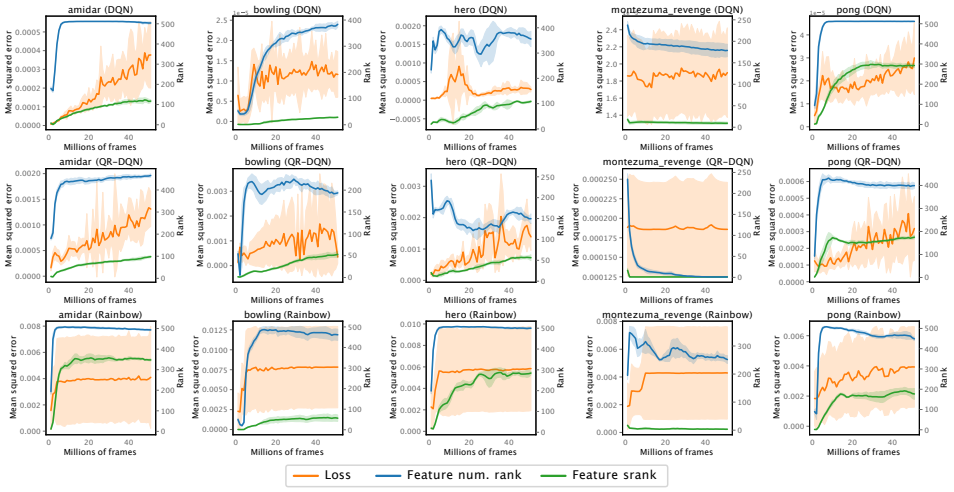


Figure D.9: Mean squared error, after 10000 training steps for the target-fitting on random network targets. We also show the corresponding feature rank of the pre-trained neural network (before target-fitting).

D.3.4 Performance

We provide full training curves for both Rainbow and Rainbow+InFeR on all games in Figures D.10 & D.11 (capped human-normalized performance), and D.12 & D.13 (raw evaluation score). We also provide evaluation performance curves for DDQN and DDQN+InFeR agents in Figure D.14.

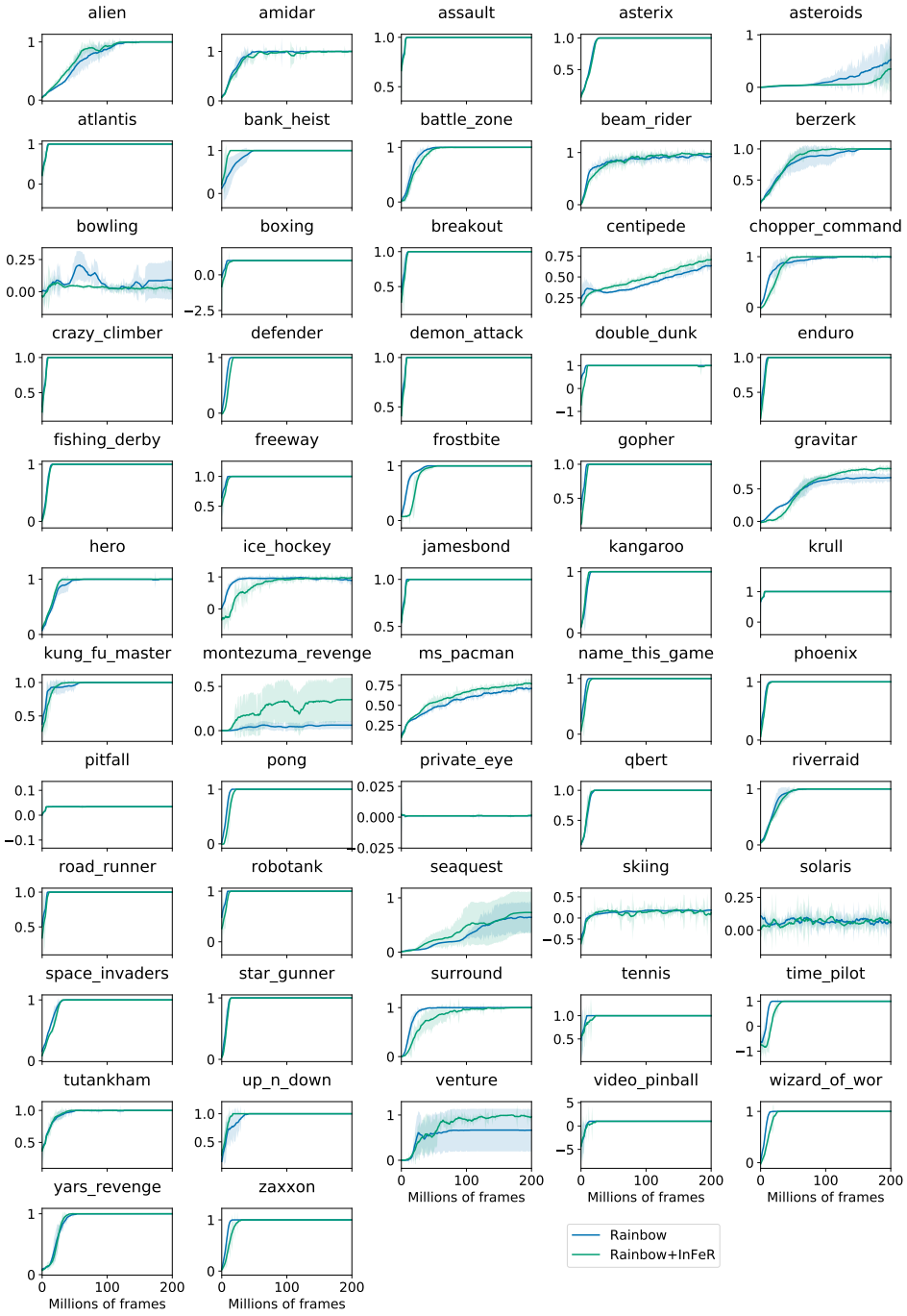


Figure D.10: Full evaluation of capped human-normalized performance on Atari benchmarks for the default Rainbow architecture.

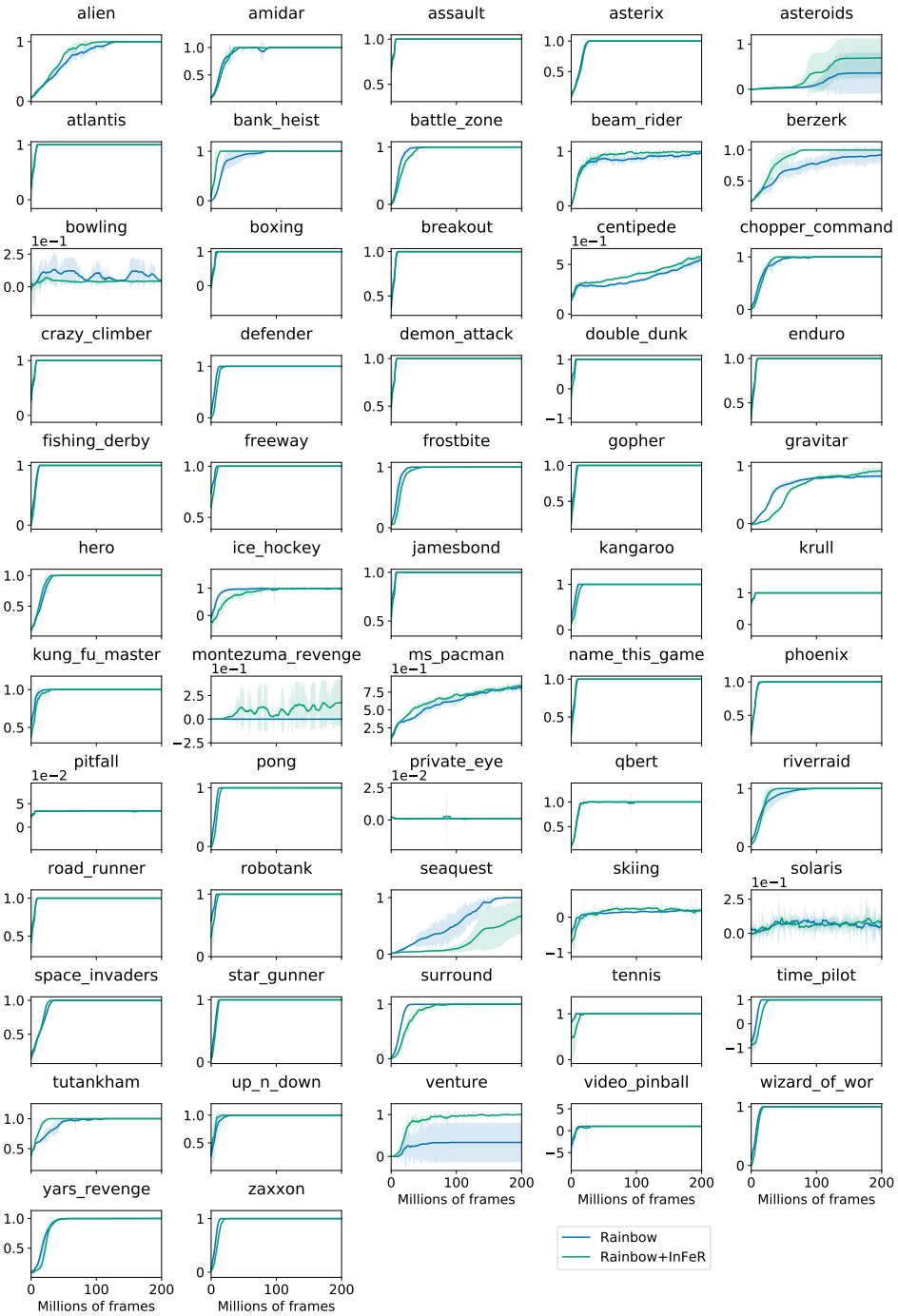


Figure D.11: Full evaluation of capped human-normalized performance on Atari benchmarks in the double-width Rainbow architecture.

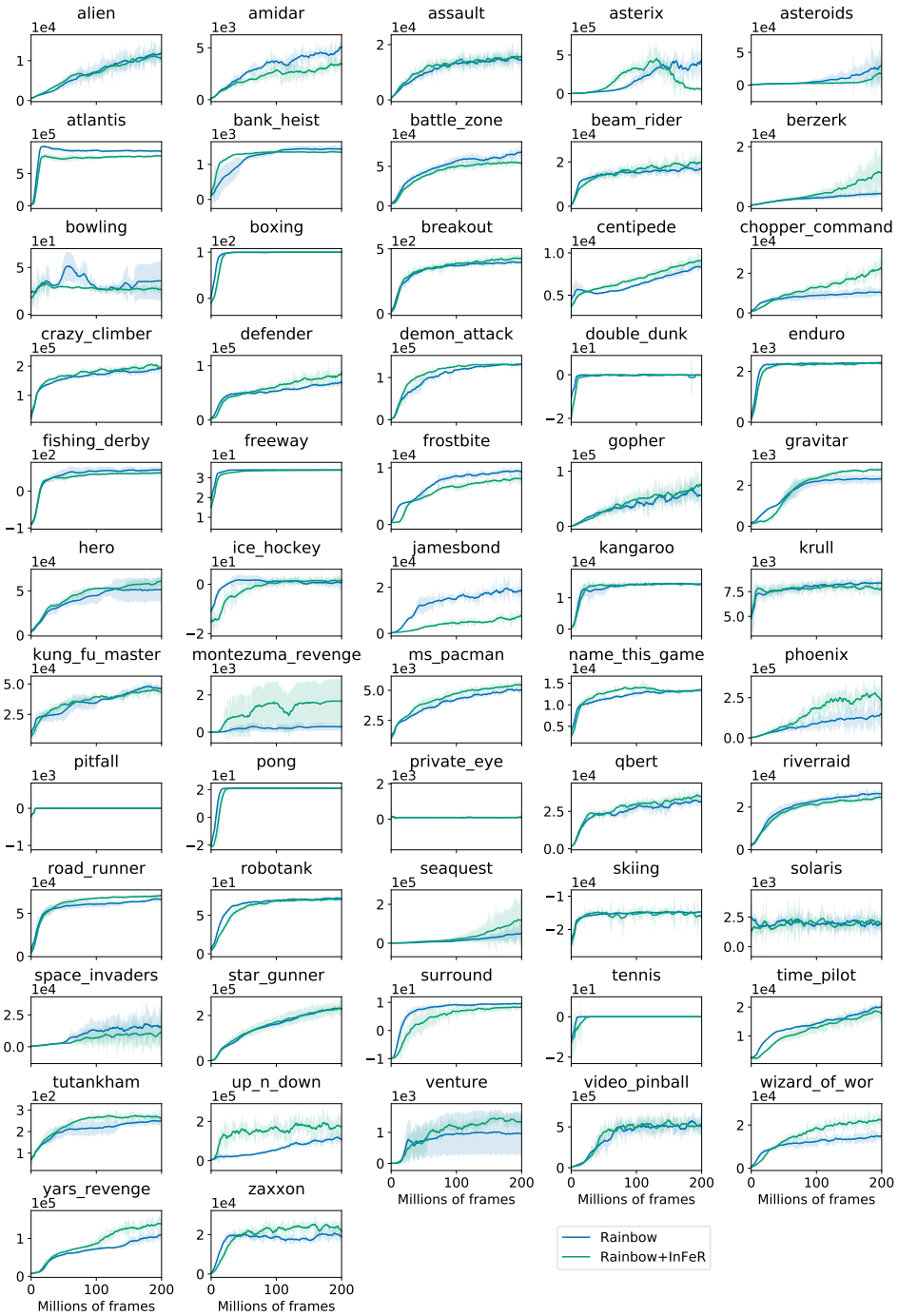


Figure D.12: Full evaluation of raw scores on Atari benchmarks for the default Rainbow architecture.

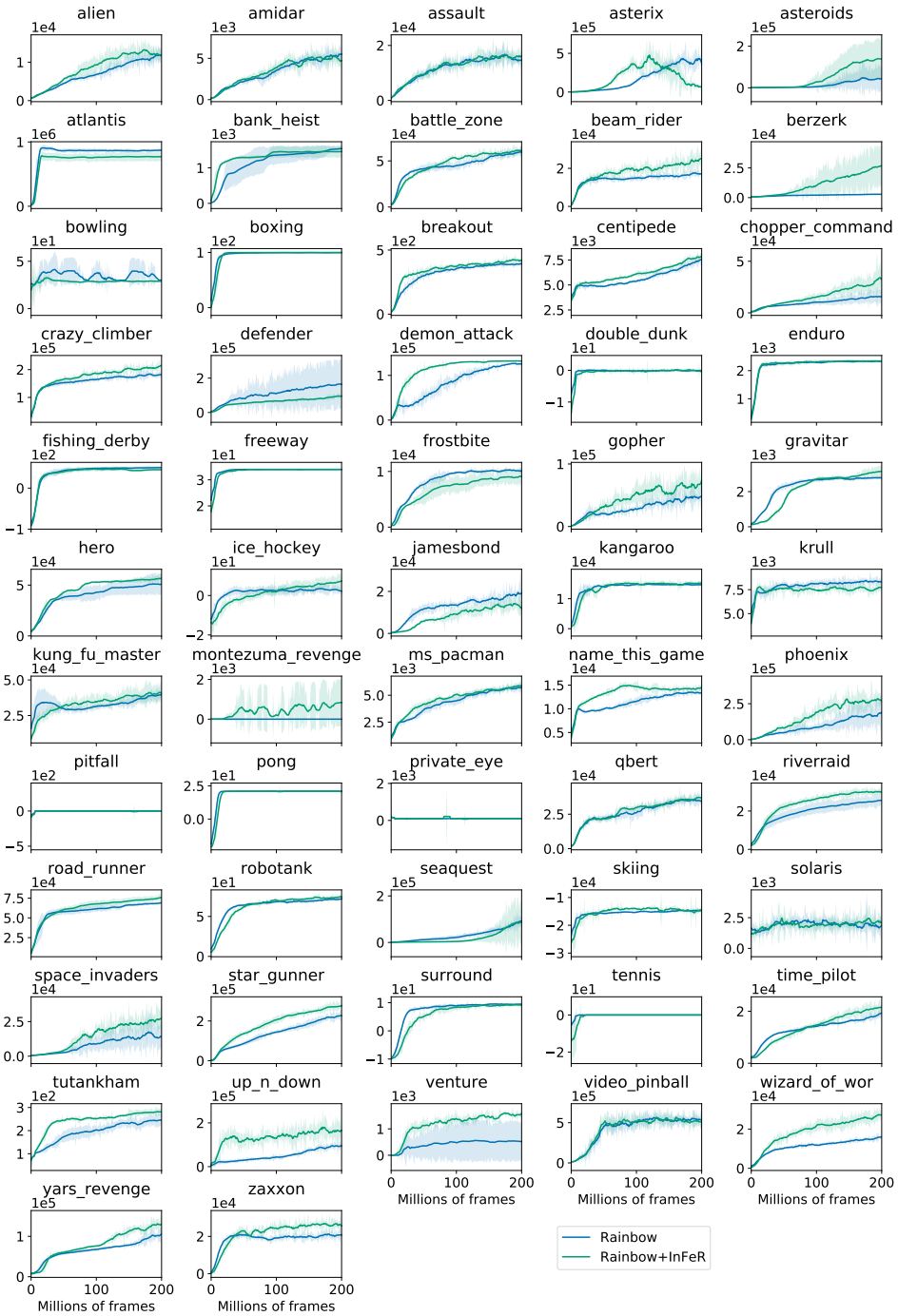


Figure D.13: Full evaluation of raw scores on Atari benchmarks for the double-width Rainbow architecture.

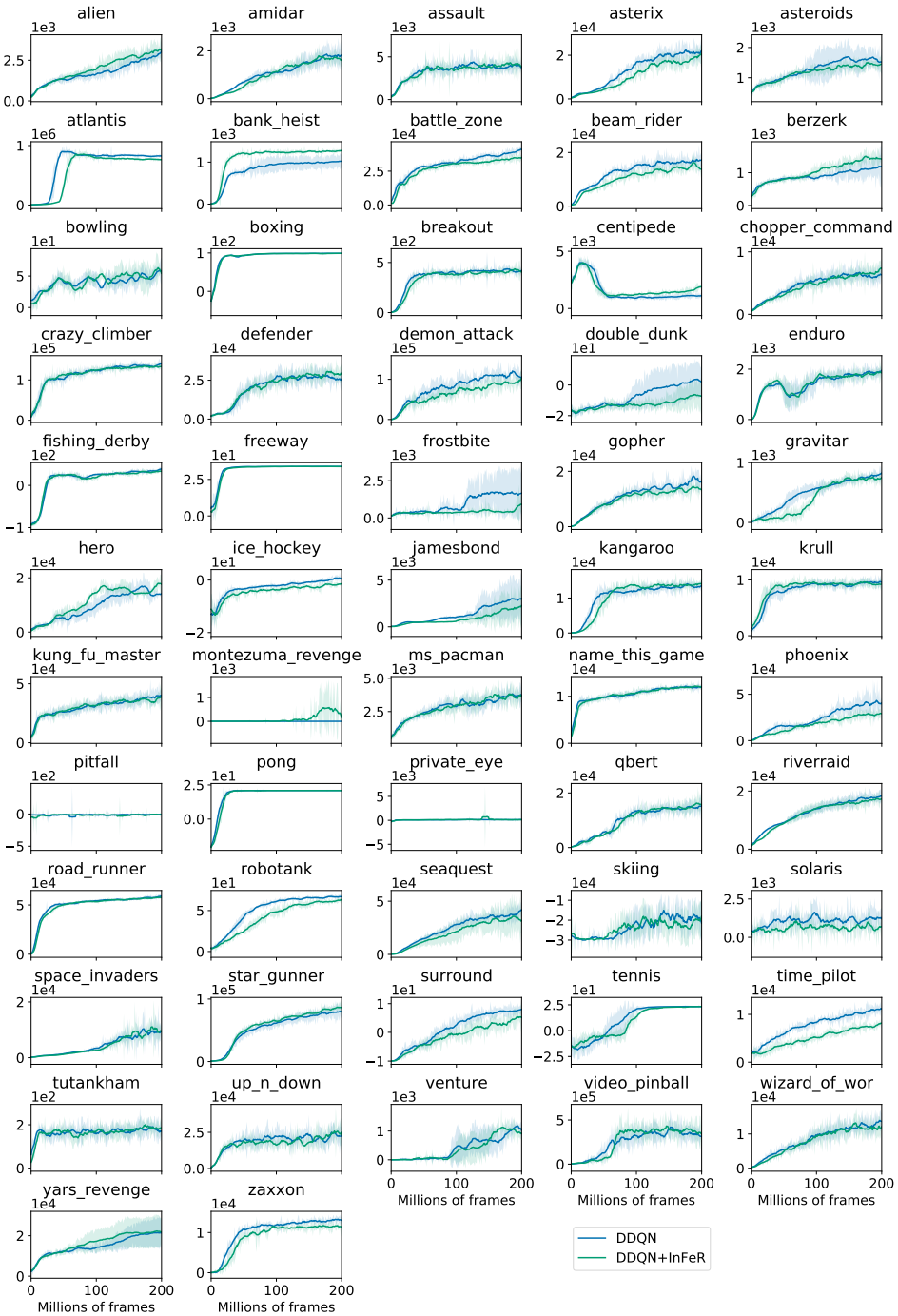


Figure D.14: Evaluations of the effect of InFeR on performance of a Double DQN agent. Overall we see a slight improvement in average performance over all games, alongside a significant improvement in Montezuma's Revenge.

E

Interference and generalization

E.1 Proofs

E.1.1 Proofs of main results

Observation 7.2.1. *Let P^π be diagonalizable, with eigenvectors $v_1, \dots, v_{|\mathcal{X}|}$ corresponding to eigenvalues $\lambda_1 > \cdots > \lambda_{|\mathcal{X}|}$, and let V_t be defined as in (7.4). Write $V_t = \sum_{i=1}^{|\mathcal{X}|} \alpha_i^t v_i$ to express the value function at time t with respect to the eigen-basis $\{v_i\}$. Then the convergence of V_t to the value function $V^\pi = \sum_{i=1}^{|\mathcal{X}|} \alpha_i^\pi v_i$ can be expressed as follows.*

$$\alpha_i^t - \alpha_i^\pi = \exp(-t(1-\gamma\lambda_i))(\alpha_i^0 - \alpha_i^\pi)$$

Proof. Recall we assume the following dynamical system

$$\partial_t V_t = -(I - \gamma P^\pi)V_t + R$$

Inducing the trajectory

$$V_t = \exp(-t(I - \gamma P^\pi))(V_0 - V^\pi) + V^\pi$$

As we assume P^π is diagonalizable, this implies that $(I-\gamma P^\pi)$ is also diagonalizable. Let $u_1, \ldots, u_n$ denote the right eigenvectors of P^π with corresponding eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$. Let $V_0 = \sum \alpha_i^0 u_i$.

$$\begin{aligned} V_t &= \sum \alpha_i^t u_i \\ &= \exp(-t(I-\gamma P^\pi))(\sum \alpha_i^0 - \alpha_i^\pi u_i) + \sum \alpha_i^\pi u_i \\ &= \sum \exp(-t(1-\gamma\lambda_i))\left(\sum(\alpha_i^0 - \alpha_i^\pi)u_i + \sum \alpha_i^\pi u_i\right) \end{aligned}$$

Now, we consider the value of $V_t - V^\pi$ along each coordinate. Note that we have not assumed an orthogonal eigenbasis, thus cannot speak directly to the norm of the projection of this difference onto the eigenspace corresponding to each eigenvector λ_k. However, treating the eigendecomposition as a basis, we can discuss how the coordinates α_i^t of the value function V_t converge with respect to this basis.

$$\begin{aligned} |V_t - V^\pi|[i] = |\alpha_i^t - \alpha_i^\pi| &= |\exp(-t(1-\gamma\lambda_i))(\alpha_i^0 - \alpha_i^\pi) + \alpha_i^\pi - \alpha_i^\pi| \\ &= |\exp(-t(1-\gamma\lambda_i))(\alpha_i^0 - \alpha_i^\pi)| = \exp(-t(1-\gamma\lambda_i))|(\alpha_i^0 - \alpha_i^\pi)| \end{aligned}$$

We conclude by noting that for large values of λ_i, the exponential term $\exp(-t(1-\gamma\lambda_i))$ will decay more slowly as a function of t than for smaller values of λ_i. Thus, these coordinates (which correspond to non-smooth functions over the state space) will converge fastest. When the eigenvectors form an orthogonal basis, as is the case for symmetric P^π, we can go further and observe that this convergence will apply to the norm of the projection of the value function into the corresponding eigenspace. Thus for symmetric P^π, we obtain the following stronger convergence result, where U_k denotes the eigenspace corresponding to the eigenvalue λ_k.

$$\|\Pi_{U_k}(V_t - V^\pi)\| = \exp(-t(1-\gamma\lambda_k))\|\Pi_{U_k}(V_0 - V^\pi)\| \tag{E.1}$$

□

Theorem 7.2.1. *Let P^π be real diagonalizable with eigenvalues $\lambda_1 > \cdots \geq \lambda_n$ and $(v_k)_{k=1}^n$ the corresponding (normalized) eigenvectors. Then for any value function V, the TD error $\mathrm{TD}(V_t) = \|V_t - T^\pi V_t\|^2$ can be bounded as*

$$\|\mathrm{TD}(V_t)\|^2 = \|T^\pi V_t - V_t\|^2 \tag{7.5}$$

$$= \|\sum(1-\gamma\lambda_i)(\alpha_i^\pi - \alpha_i^t)(v_i)\|^2 \tag{7.6}$$

$$\leq \sum_{i=1}^n (\alpha_i^\pi - \alpha_i^t)^2(1-\gamma\lambda_i)^2 \tag{7.7}$$

with equality when P^π has orthogonal eigenvectors.

Let $V_0 = \sum \alpha_i v_i$. Then, letting V_t be defined as in Equation 7.4.

$$\mathrm{TD}(V_t) \leq \sum_{i=1}^{n} \exp(-2t(1-\gamma\lambda_i))(\alpha_i^\pi - \alpha_i^0)^2(1-\gamma\lambda_i)^2 \,. \tag{E.2}$$

Proof. By our assumption on the diagonalizability of P^π, we can leverage the previous result on the coordinates of V_t.

$$V_t - V^\pi = \sum \exp(-t(1-\gamma\lambda_i)) \left(\sum (\alpha_i^0 - \alpha_i^\pi) u_i \right)$$

We then bound the TD error as follows.

$$\begin{aligned} \|V_t - \gamma P^\pi V_t - R\|^2 &= \|V_t - \gamma P^\pi V_t + \gamma P^\pi V^\pi - \gamma P^\pi V^\pi - R\| \\ &= \|V_t - \gamma P^\pi V^\pi - R - \gamma P^\pi (V_t - V^\pi)\| \end{aligned}$$

Since $V^\pi = R + \gamma P^\pi V^\pi$, we obtain the following.

$$\begin{aligned} &= \|(I - \gamma P^\pi)(V_t - V^\pi)\|^2 \\ &= \|\sum (1-\gamma\lambda_k)(\alpha_i^t - \alpha_i^\pi) u_i\|^2 \\ &\leq \sum (\alpha_i^\pi - \alpha_i^t)^2 (1-\gamma\lambda_i)^2 \end{aligned}$$

The remainder follows a straightforward substitution. □

Observation 7.2.2 (Second-order dynamics)**.** *Let θ_t be defined by the discrete-time system (7.11) with step size α. Let $f_1(\theta)$ be defined as in (7.12). Let $\tilde{\theta}_t$ denote the trajectory obtained by following the dynamics:*

$$\partial_t \tilde{\theta}_t = f(\tilde{\theta}_t) + \frac{\alpha}{2} f_1(\tilde{\theta}_t) \,. \tag{7.13}$$

Then we have $\theta_n \approx \tilde{\theta}_{n\alpha} + O((n\alpha)^3)$, where $\tilde{\theta}_{n\alpha}$ denotes the value of $\tilde{\theta}_t$ at time $t = n\alpha$.

Proof. While our prior analysis has considered the continuous time system $\tilde{\theta}_t$, this does not perfectly approximate the discrete system θ_t. When a fixed step size is used, the first-order continuous-time approximation accrues error roughly proportional to αt. We then follow the procedure of Barrett and Dherin [2021], applying a Taylor

expansion to the evolution of $\tilde{\theta}_t$ with respect to time. We will use the notation $\tilde{\theta}(t)$ to denote the explicit dependence of $\tilde{\theta}$ as a function of time.

$$\tilde{\theta}(\alpha t) = \tilde{\theta}(0) + \sum \frac{(\alpha t)^n}{n!}\theta^{(n)}(0) \tag{E.3}$$

$$= \tilde{\theta}(0) + \alpha t f(\tilde{\theta}(0)) + \frac{(\alpha t)^2}{2}\nabla_\theta f \cdot f(\tilde{\theta}(0)) + O(\alpha^3) \tag{E.4}$$

$$= \tilde{\theta}(0) + \alpha t f(\tilde{\theta}(0)) + \frac{(\alpha t)^2}{2} f_1(\theta(0)) + O(\alpha^3) \tag{E.5}$$

Relating this back to the discrete system θ_t

$$\theta_1 = \theta_0 + \alpha f(\theta_0) = \tilde{\theta}(0) + \alpha f(\tilde{\theta}(0)) \tag{E.6}$$

$$\theta_1 = \tilde{\theta}(1) - \frac{\alpha^2}{2} f_1(\tilde{\theta}(0)) + O(\alpha^3) \tag{E.7}$$

Thus, the system $\partial_t \check{\theta}_t = f(\check{\theta}_t) + \alpha^2/2 f_1(\check{\theta}_t)$ satisfies

$$\theta_1 = \check{\theta}(1) + O(\alpha^3) \tag{E.8}$$

We begin by observing that $\nabla_\theta \|V_\theta - \square T V_\theta\|^2 = (V_\theta - T V_\theta) \cdot \nabla_\theta V_\theta = f(\theta)$.

$$\theta = \theta_0 + \alpha n f(\theta_0) + (\alpha n)^2/2 \nabla_\theta f(\theta_0) \cdot f(\theta_0) + O((\alpha n)^3) \tag{E.9}$$

$$= \theta_0 + \alpha n f(\theta_0) + \frac{(\alpha n)^2}{2} f_1(\theta_0) + O((n\alpha)^3) \tag{E.10}$$

We then express $f_1(\theta)$ as follows.

$$f_1(\theta_0) = \nabla_\theta [f(\theta_0)] \cdot [f(\theta_0)] \tag{E.11}$$

$$= [\nabla_w^2 V_\theta \cdot ((\gamma P^\pi - I)V_\theta + r) + \nabla_\theta V_\theta \cdot ((\gamma P^\pi - I)\nabla_\theta V_\theta)][f(\theta)] \tag{E.12}$$

$$= [\nabla_\theta^2 V_\theta \cdot ((\gamma P^\pi - I)V_\theta + r) + \nabla_\theta V_\theta \cdot \nabla_\theta V_\theta][f(\theta)] \tag{E.13}$$

$$+ \gamma [\nabla_\theta V_\theta P^\pi \nabla_\theta V_\theta][f(\theta)] \tag{E.14}$$

Noting that the left hand side term is equal to the gradient of the gradient norm penalty for the stop-gradient version of the TD regression problem, we simplify as follows:

$$= \frac{1}{2}\nabla_\theta \|\nabla_\theta \frac{1}{2}\|V_\theta - \square T^\pi V_\theta\|^2\|^2 + \gamma [\nabla_\theta V_\theta \cdot P^\pi \cdot \nabla_\theta V_\theta][f(\theta)] \tag{E.15}$$

We note that, unlike in the stochastic gradient descent setting, f_1 does not correspond to a gradient of any function. Instead, it corresponds to the second-order we would get for a frozen target, which corresponds to a gradient norm penalty, plus a term that measures the alignment of the gradients at each state and its expected successor. Intuitively, both of these terms minimize the 'variance' in the loss induced by noisy, discrete gradient steps. The flatter loss surfaces induced by the gradient norm penalty will naturally lead to greater robustness to parameter perturbations. The gradient alignment term reflects the observation previously that non-smooth functions contribute the most to the TD error, and so encourages the first-order gradient effects on successive states to move in a similar direction.

We note that this final observation seems to be at odds with the tendency for TD learning to encourage more tabular updates. Why would a second-order correction term which promotes flat minima and gradient alignment result in tabular updates? To answer this, we point to the tendency of TD targets to converge along. the non-smooth components of the value function first. We are therefore faced with finding a flat region of parameter space to fit a discontinuous function. A representation which succeeds at this will benefit from minimizing interference between states, as the gradients for one transition will be on average uncorrelated with even nearby other states. The gradient alignment penalty suggests that, while the implicit regularization will prefer flat minima, smooth interference patterns which move other states in a similar direction to the current state will be penalized less than non-smooth directions. □

Corollary E.1.1. *The second-order dynamics push features towards precisely the worst direction w.r.t. stability. I.p. looking at the set of positive definite representations introduced by Ghosh and Bellemare [2020] we see*

$$\{v : v^\top P^\pi v < \gamma^{-1}\|v\|_\Xi\} \tag{E.16}$$

whereas the optimal gradients for the second order term implicitly solve the following optimization problem

$$\min \mathbb{E}_{x\sim\eta(x)}[g(x)^\top g(x) - \gamma g(x)^\top (P^\pi g)(x)] \tag{E.17}$$

Theorem 7.2.2. *Let K be a kernel and π a fixed policy in an MDP with finite state space X. Let $X_{\text{train}} \subset \mathcal{X}$ be the set of states in the support of π, $X_{\text{test}} = \mathcal{X} \setminus X_{\text{train}}$, and let V_t be a value trajectory obtained by applying kernel semi-gradient updates to some initial value function V_0 with kernel K. Let K_{all} be defined as*

$$K_{\text{all}} = K(X_{\text{train}}, X_{\text{train}}) \oplus K(X_{\text{test}}, X_{\text{train}}) \tag{7.9}$$

Then the trajectory of V_t on the entire state space X will be as follows,

$$\partial_t V_t(X) = (K_{\text{all}})[(T^\pi V_t - V_t)(X_{\text{train}})] \,. \tag{7.10}$$

Proof. We leverage the dynamics $\partial_t V_t = K(X, X)\nabla_\theta V_\theta \cdot ((\gamma P^\pi - I) + r)$ and follow the derivation of Section 5 of Jacot et al. [2018]. □

We can develop intuitions for the kernel gradient descent setting by considering the special case of linear function approximation, where $K(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$ for some feature map ϕ. For the moment, we will define Φ to be a matrix consisting of features for every state in the state space X (i.e. we update all states in the mdp at once). We then obtain

$$\partial_t \mathbf{w}_t = \alpha \Phi^\top (R^\pi + \gamma P^\pi \Phi \mathbf{w}_t - \Phi \mathbf{w}_t) \,. \tag{E.18}$$

We can express the evolution of the value function constructed by multiplication of Φ and w as follows.

$$\begin{aligned}
\partial_t V_t &= (\partial_w V_t)^\top \partial_t w_t = \Phi \partial_t w_t && \text{(E.19)} \\
&= -\Phi(\Phi^\top (I - \gamma P^\pi)\Phi) w && \text{(E.20)} \\
&= -\Phi\Phi^\top (I - \gamma P^\pi) V_t && \text{(E.21)} \\
&= -K(I - \gamma P^\pi) V_t && \text{(E.22)}
\end{aligned}$$

We further consider the dynamics of the value function on inputs outside of the set of states on which the Bellman updates are computed as follows.

$$\begin{aligned}
\partial_t V_t(x_{\text{test}}) &= (\partial_w V_t(x_{\text{test}}))^\top \partial_t w_t && \text{(E.23)} \\
&= -\phi(x_{\text{test}})^\top \Phi^\top (I - \gamma P^\pi) V_t && \text{(E.24)} \\
&= -K(x_{\text{test}}, X_{\text{train}}) K(X_{\text{train}}, X_{\text{train}})^{-1} \partial_t V_t && \text{(E.25)}
\end{aligned}$$

We now lift the assumption that all states are updated. In this more general kernel gradient descent setting, we let K be a kernel as before, with $\tilde{K} = K(X_{\text{train}}, X_{\text{train}})$ and $\kappa_{x_{\text{test}}} = K(x_{\text{test}}, X_{\text{train}})$. We then obtain the following dynamics

$$\partial_t V_t(x_{\text{test}}) = \kappa_{x_{\text{test}}} \tilde{K}^{-1} \partial_t V_t(X_{\text{train}}) \tag{E.26}$$

In particular, this results in the following trajectory.

$$V_t(x_{\text{test}}) = V_0(x_{\text{test}}) + \kappa_{x_{\text{test}}} \tilde{K}^{-1}[V_t(X_{\text{train}}) - V_0(X_{\text{train}})] \tag{E.27}$$

An interesting case study occurs when we consider, e.g., off-policy evaluation where the bootstrap targets used in the TD updates may not have been visited by the agent during training. This will be the case in many offline RL problems, where the action that would be selected by the policy we seek to evaluate was not taken by the behaviour policy, and so the agent leverages bootstrap targets which are not updated as part of the training set of states. In such cases, we will decompose the state space as $X = X_{\text{train}} \oplus X_{\text{test}}$. The dynamics we get in this case look quite different from standard kernel regression, as the dynamics of the training states will depend on the predictions on the 'test' states. To condense notation, we will use $T^\pi V_t$ to refer to an application of the Bellman operator $V_t \mapsto \gamma P^\pi V_t + R^\pi$.

$$\partial_t V_t(X_{\text{train}}) = \Phi_{\text{train}} \Phi_{\text{train}}^\top ((T^\pi V_t)(X_{\text{train}}) - V_t(X_{\text{train}})) \tag{E.28}$$

$$\partial_t V_t(X_{\text{test}}) = \Phi_{\text{test}} \Phi_{\text{train}}^\top ((T^\pi V_t)(X_{\text{train}}) - V_t(X_{\text{train}})) \tag{E.29}$$

We note that $(T^\pi V_t)(X_{\text{train}})$ depends on both $V(X_{\text{train}})$ and $V(X_{\text{test}})$ due to the application of the Bellman operator T^π. We thus end up with the following joint system.

$$\partial_t V_t(X_{\text{train}} \oplus X_{\text{test}}) = \Phi_{\text{test}} \Phi_{\text{train}}^\top ((T^\pi V_t)(X_{\text{train}}) - V_t(X_{\text{train}})) \oplus \Phi_{\text{train}} \Phi_{\text{train}}^\top ((T^\pi V_t)(X_{\text{train}}) - V_t(X_{\text{tr}} \tag{E.30}$$

$$\partial_t V_t(X_{\text{train}} \oplus X_{\text{test}}) = (\Phi_{\text{test}} \oplus \Phi_{\text{train}}) \Phi_{\text{train}}^\top ((T^\pi V_t)(X_{\text{train}}) - V_t(X_{\text{train}})) \tag{E.31}$$

Using a non-standard notation of $K_1 \oplus K_2 := X \mapsto K_1(X) \oplus K_2(X)$, we can then rewrite the above in terms of the dot product kernel $K(x, x')$ as follows.

$$\partial_t V_t(X_{\text{all}}) = (\tilde{K} \oplus \kappa_{x_{\text{test}}})[(T^\pi V_t - V_t)(X_{\text{train}})] \tag{E.32}$$

We emphasize that while this at first looks as though the dynamics are independent of the value $V_t(X_{\text{test}})$, this is an artefact of the Bellman operator notation $(T^\pi V_t)(X_t)$, which hides the dependence of the Bellman targets $T^\pi V_t$ on X_{test}. In particular, we can write $(T^\pi V_t)(X_t) = \Pi_{X_{\text{train}}}[\gamma P^\pi V_t(X_{\text{train}} \oplus X_{\text{test}}) + R^\pi]$, which makes this dependence more explicit but is less succinct.

E.2 Experiment details

E.2.1 Estimation of update rank

To estimate the update rank of an agent, we sample k transitions from the agent's replay buffer and compute the matrix $A(\theta)$ as described in Section 7.3. We use the agent's current optimizer state and its current parameters in this computation. We then take the singular value decomposition of A to obtain k singular values $S = \{\sigma_1, \ldots, \sigma_k\}$. We then threshold using the numerical approach taken in prior works [Maddox et al., 2020], and compute the size of the set $S_\epsilon = \{\sigma \in S : \sigma > \epsilon \max(S)\}$. This allows us to ignore directions of near-zero variation in the update matrix. In practice, we use $\epsilon = 0.1$.

Because the Q-functions learned by value-based deep RL agents are vector- rather than scalar-valued functions of state, and our estimator depends on an 2-dimensional update matrix, we must make a choice on how to represent the change in the state-value function. We considered taking the maximum over actions, the mean over actions, selecting a fixed action index, and selecting the action taken in the transition on which the update was computed, and found that both choices produced similar trends. In all evaluations in this paper, Q-functions are reduced using the max operator. We apply the same approach for distributional agents by taking the expectation over the distribution associated with each state-action pair.

To evaluate the policy-based agents, whose outputs correspond to distributions over actions, we compute the norm of the difference in the output probability distributions for each state in lieu of taking the difference of output values. I.e., the entry $A_{i,j} = \|p_\theta(x_j) - p_{\theta_i}(x_j)\|$, where the discrete probability distribution p_θ is taken as a vector.

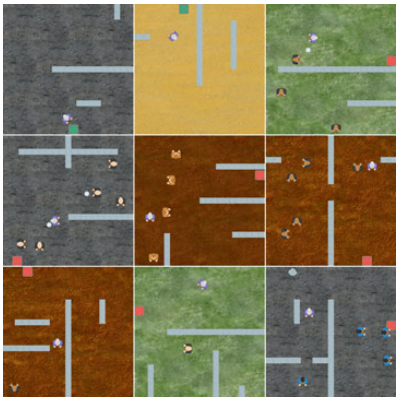


Figure E.1: Example levels from the dodgeball environment.

E.2.2 ProcGen

The ProcGen benchmark consists of sixteen procedurally generated environments. Each environment consists of a set of randomly generated levels, of which a fixed subset are used for training and a disjoint subset are used for evaluation. Levels differ superficially in their observations and initial sprite layouts but retain the same underlying structure, as can be seen in Figure E.1. The observation space is a box space with the RGB pixels the agent sees in a numpy array of shape (64, 64, 3).

Our PPO and DAAC agents use the same hyperparameters and implementation as is provided by Raileanu and Fergus [2021]. Our behaviour cloning objective minimizes the KL divergence between the distillation agent the pretrained agent's policies, with an entropy bonus equal to that used to train the original PPO agent.

E.2.3 Atari

We additionally perform evaluations on environments from the Atari benchmarks. Due to computational constraints, we consider only a subset of the entire benchmark. We obtain a mixture of easy games, such as pong and boxing, and more challenging games like seaquest, where we measure difficulty by the time it takes for the agent to meet human performance. For some experiments, we used the sparse-reward environment Montezuma's Revenge.

In our distillation experiments, we train the original agent for 50M frames using ϵ-greedy exploration with $\epsilon = 0.1$, and train the distillation agents for a

number of updates equivalent to 10M frames of data collected online. We base our implementation off of the open-source implementations in Ostrovski et al. [2021].

For our behaviour cloning objective, we use the same architecture as is used for DQN, but feed the final layer of actions into a softmax to obtain a probability distribution over actions, which we denote as $P_\theta(a|x)$. Given a state-action pair taken by the target agent, we implement the following behaviour cloning loss for distillation

$$\ell(\theta, x_i, a_i) = -\log P_\theta(a_i|x_i) - 0.1H(P_\theta(\cdot|x_i)) \tag{E.33}$$

where H denotes the entropy of a distribution. We use a replay capacity of 1e6 and allow the pre-trained agent to collect additional data during distillation to further increase the training set size of the distilled agents.

E.3 Additional numerical evaluations

We provide additional numerical evaluations to provide insight into the theoretical results of Section 7.2.

E.3.1 Fourier analysis

We begin by studying the Fourier decomposition of value and reward functions in popular Atari domains by treating the value function as a function of *time* rather than as a function of *observations*. In this sense, the Fourier decomposition is measuring the continuity of the value function with respect to time and so is a closer approximation of the notion of smoothness we focus on in Section 7.2.1. We show our evaluations in Figure E.2.

E.3.2 Kernel gradient descent

We include an illustration of the kernel gradient descent dynamics described in Section 7.2.2 in Figure E.3. We run our evaluations using a radial basis function (RBF) kernel of varying lengthscale, with shorter lengthscales corresponding to weaker generalization between states. While the shorter lengthscale corresponds to more stable learning dynamics and better fitting of the value function on the

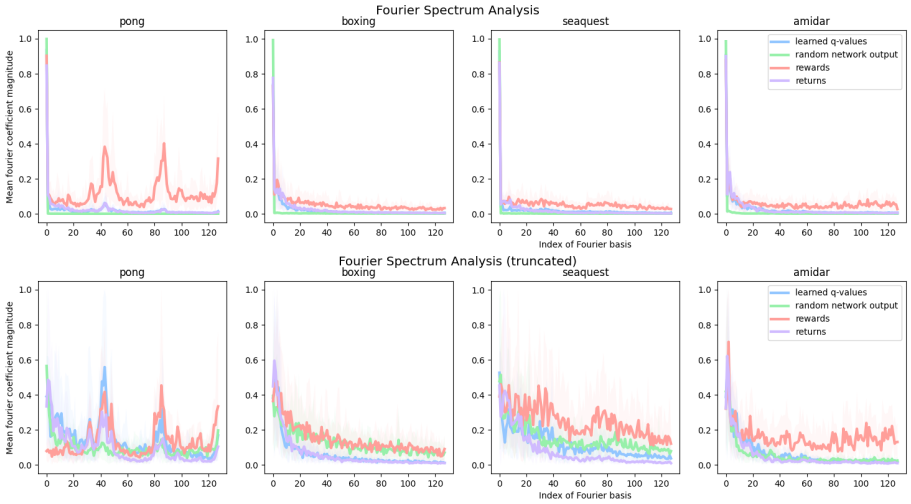


Figure E.2: Fourier decomposition of Atari value functions when viewed as a function of time. We sample k consecutive states from the replay buffer and compute the predicted value on each state (fixing an arbitrary action) to get a function $V : \{1, \ldots, k\} \to \mathbb{R}$. We then compute the Fourier decomposition of this function. The top row shows indices $k = 0 \ldots 50$, while the bottom row omits the $k = 0$ index (the constant function) to better illustrate the rate of decay of the spectrum of each function.

training set, it also induces greater value approximation error on the test states. In contrast, the longer lengthscales result in better generalization to novel test states under Monte Carlo dynamics, but result in divergence for large values of γ.

Additionally, as promised in Section 7.2.2, we illustrate the role of smooth eigenfunctions in generalization in Figure E.4. To produce this figure, we randomly generate an unweighted graph and then construct an MDP whose dynamics correspond to a random walk on this graph. We consider the generalization error of a kernel regression process where the kernel K_S is of the form $K_S(x, y) = \sum_{i \in S} v_{\lambda_i}(x) v_{\lambda_i}(y)$ for some $S \subseteq \text{spec}(P^\pi)$. In the right-hand-side plot of Figure E.4, we set $S = \{1, \ldots, 20\}$, so that our analysis concentrates on smooth eigenfunctions. We then consider the generalization error of this smooth kernel when we only regress on a subset of the state space selected uniformly at random[1]. We study the effect of varying the size of this set, i.e. the fraction of states in the training set, in Figure E.4, in order to quantify the degree to which additional information about the value function translates to improved generalization. We consider three

[1]Because the MDP-generating process is invariant to permutations of the state indices, we sample the indices $\{1, \ldots, \lfloor |\mathcal{X}| \times \text{trainingfraction} \rfloor\}$, and average over randomly generated MDPs.

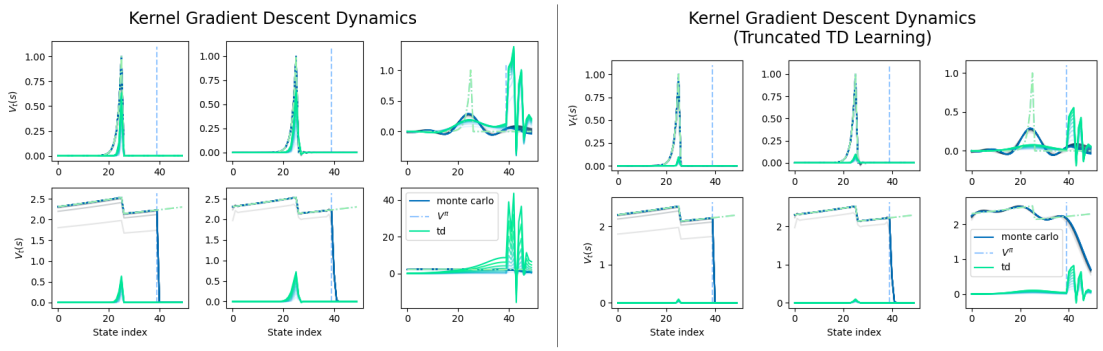


Figure E.3: Numerical evaluations of kernel gradient descent with an RBF kernel. The MDP in question is a "circle MDP" whose states are integers $n \in \{1, \ldots, 50\}$. We assume the agent is 'trained' on states 1 to 40, and does not perform value function updates on the final ten states, use the policy which always takes the agent from state n to $n+1$ mod 50, and set a single reward at state 25. Each row corresponds to a different value of the discount factor γ: the top corresponds to $\gamma = 0.5$, and the bottom to $\gamma = 0.99$. Each column corresponds to the lengthscale which parameterizes the kernel, going left to right: 0.01, 1.0, and 100. The left hand side and right hand side are distinguished by the number of update steps which the TD dynamics are evaluated for. The LHS runs TD for only 20 steps, while the RHS runs it for 100 steps. MC updates are run for 1500 steps on both figures. We see that for $\gamma = 0.99$, the larger-lengthscale kernel predictions diverge under TD dynamics, though not Monte Carlo. The Monte Carlo dynamics further nicely illustrate the trade-off between generalizing out of the training set and ability to fit the discontinuities of the value function on the training set. The larger lengthscale has lower MSE from the value function on the test set, but fails to fit the discontinuity of the value function at the reward state. Meanwhile, the smaller lengthscales easily fit the value function on the training set but predict zero for all over states.

regression problems: regression on V^π, regression on the projection of V^π onto the span of $T = \{v_1, \ldots, v_{20}\}$, and $B = \{v_{n-19}, \ldots, v_n\}$. Unsurprisingly, we see that the smooth kernel is able to improve its generalization performance as the size of the training set increases when it is set to regress V^π or $\Pi_T V^\pi = V_T^\pi$. However, when the kernel regresses only on the projection of V^π onto the non-smooth eigenvectors, we do not see a benefit of adding additional training points: because there is no information about the smooth components of the function in the targets, adding additional data points will not help to improve regression accuracy. The left hand side of the figure shows similarly that fitting local information in the form of n-step returns for small n also does not provide the kernel with sufficient information for it to be able to extrapolate and improve its generalization error as the size of the training set increases.

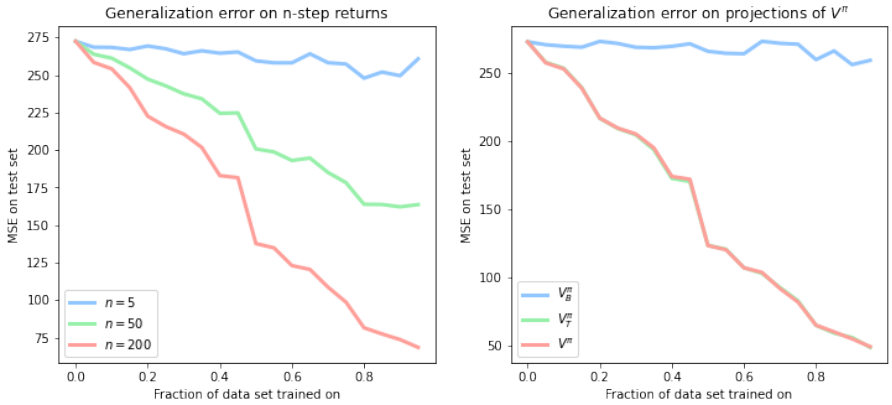


Figure E.4: Generalization of predicted function under kernel regression using n-step return targets evaluated on a random subset of states (left), and projecting value function onto top or bottom eigenvectors of P^π (right). We see a similar trend where for larger n (corresponding to smoother targets), the kernel regression method generalizes better with increasing dataset sizes. For smaller n and for the projection of V^π onto non-smooth eigenvectors, adding additional data points does not improve generalization performance.

E.4 Additional empirical results

E.4.1 Additional value distillation results

We consider three different types of regression to the outputs of the pre-trained network, along with two more traditional bootstrapping methods for offline RL. `Q-regression` regresses the outputs of the distilled network to those of the pre-trained network for every action. `qa-regression` only does q-value regression on the action taken by the pre-trained agent. `adv-regression` regresses on the advantage function (computed as the q-value minus the mean over all actions) given by the pre-trained agent; `qr` does quantile regression q-learning on the offline data; `double-q` performs a standard double q-learning update on the offline data.

We find that all of these methods obtain an initial update rank significantly below that of the pre-trained network when they begin training, which increases over time. Regression to the advantages obtains a significantly lower update rank than any other method, suggesting that the advantage function may be much smoother than the action-value function. With respect to performance on the original environment, we see that the methods which use all action values at every update obtain significantly higher performance than those which only update a

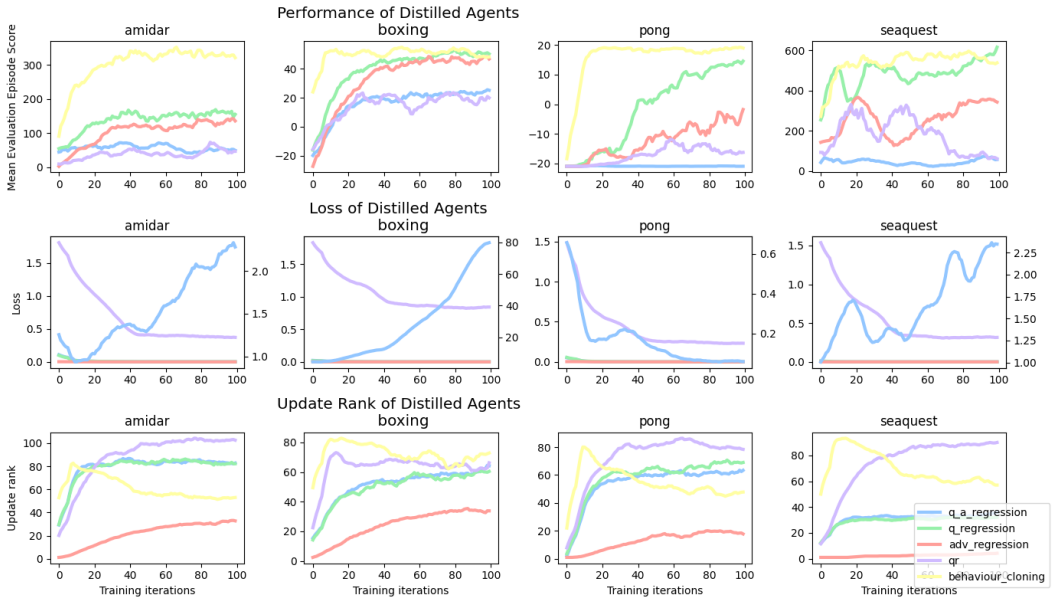


Figure E.5: Results from post-training distillation on a variety of objectives. We note that advantage regression tends to exhibit the lowest update rank, with the qr agent tending to exhibit the highest update rank and the q-regression objectives falling somewhere in between. Because the behaviour cloning objective minimizes a cross-entropy loss rather than a regression loss, further investigation is required to understand how the trajectory of its update dimension differs from those of the regression objectives.

single action at a time. This improvement in performance is not mediated by an auxiliary task effect or an increase in the network's ability to distinguish states: the advantage regression network attains low update rank but high performance, while the qr-regression task provides a great deal of information to the representation but is not competitive with the q-regression network.

E.4.2 More detailed update trajectories

We include a more detailed view of the update matrices obtained by DQN and C51 agents during the first 7 million frames of training, roughly 5% of the training budget, in Figure E.6. We see that even early in training, the DQN and C51 agents both exhibit significant overfitting behaviour. Note that states are sampled uniformly at random from the replay buffer, and then assigned an index based on the output of a clustering algorithm to improve readability of the figures.

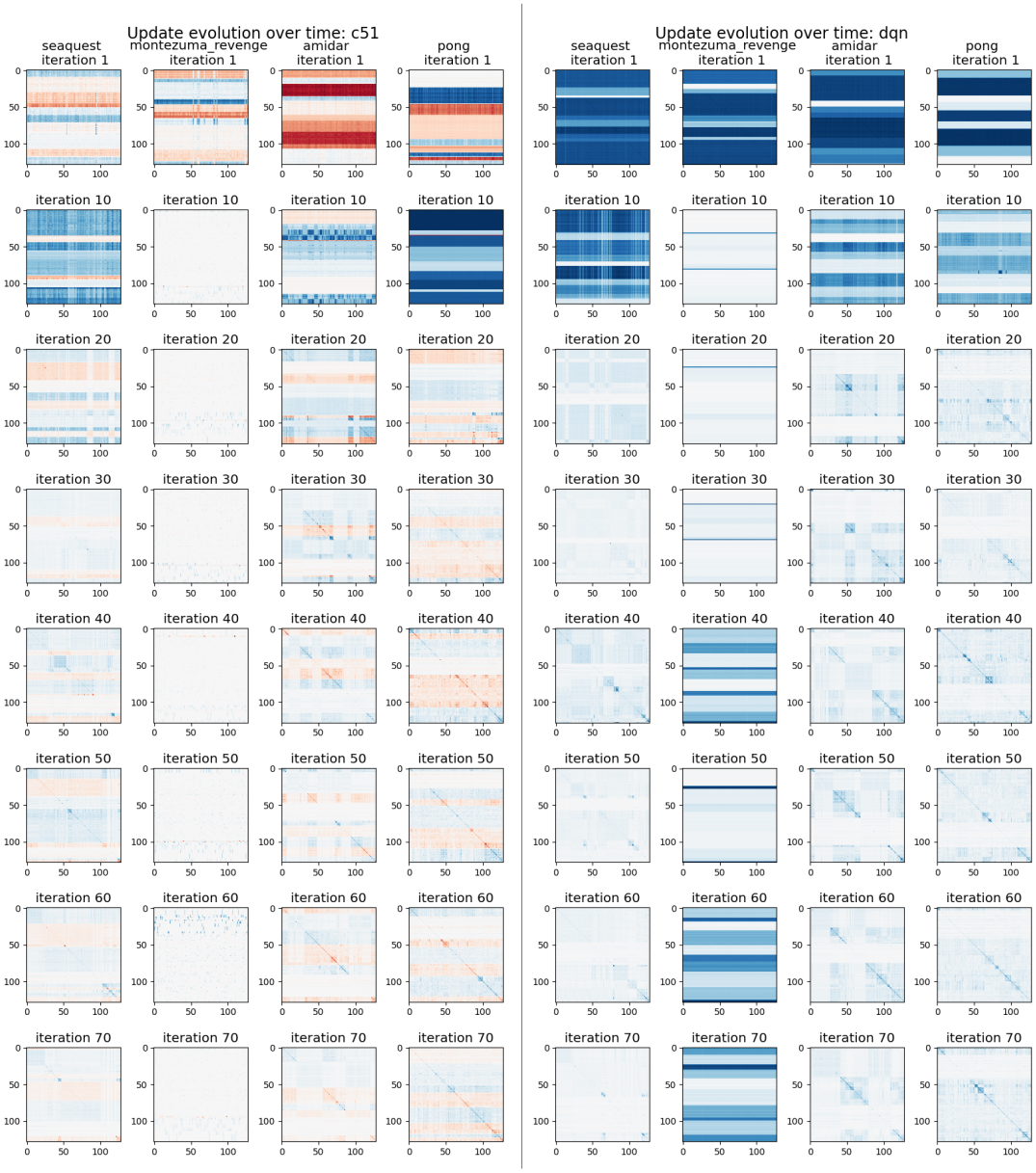


Figure E.6: Update matrices for distributional and DQN agents on four games from the Atari suite, chosen to represent a range of reward densities and difficulties. Each iteration corresponds to 1e5 training frames.

F

Generalization across environments

F.1 Implementation details

F.1.1 Model learning: rich observations

For the model learning experiments we use an almost identical encoder architecture as in Tassa et al. [2018], with two more convolutional layers to the convnet trunk. Secondly, we use `ReLU` activations after each convolutional layer, instead of `ELU`. We use kernels of size 3×3 with 32 channels for all the convolutional layers and set stride to 1 everywhere, except of the first convolutional layer, which has stride 2. We then take the output of the convolutional net and feed it into a single fully-connected layer normalized by `LayerNorm` [Ba et al., 2016]. Finally, we add `tanh` nonlinearity to the 50 dimensional output of the fully-connected layer.

The decoder consists of one fully-connected layer that is then followed by four deconvolutional layers. We use `ReLU` activations after each layer, except the final deconvolutional layer that produces pixels representation. Each deconvolutional layer has kernels of size 3×3 with 32 channels and stride 1, except of the last layer, where stride is 2.

The dynamics and reward models are all MLPs with two hidden layers with 200 neurons each and `ReLU` activations.

F.1.2 Reinforcement learning

For the reinforcement learning experiments we modify the Soft Actor-Critic PyTorch implementation by Yarats and Kostrikov [2020] and augment with a shared encoder between the actor and critic, the general model f_s and task-specific models f_η^e. The forward models are multi-layer perceptions with ReLU non-linearities and two hidden layers of 200 neurons each. The encoder is a linear layer that maps to a 50-dim hidden representation. We also use L1 regularization on the S latent representation. We add two additional dimensions to the state space, a spurious correlation dimension that is a multiplicative factor of the last dimension of the ground truth state, as well as an environment id. We add Gaussian noise $\mathcal{N}(0, 0.01)$ to the original state dimension, similar to how Arjovsky et al. [2019] incorporate noise in the label to make the task harder for the baseline.

Soft Actor Critic (SAC) Haarnoja et al. [2018] is an off-policy actor-critic method that uses the maximum entropy framework to derive soft policy iteration. At each iteration, SAC performs soft policy evaluation and improvement steps. The policy evaluation step fits a parametric soft Q-function $Q(x_t, a_t)$ using transitions sampled from the replay buffer $\mathcal{D}$ by minimizing the soft Bellman residual,

$$J(Q) = \mathbb{E}_{(x_t, x_t, r_t, x_{t+1}) \sim \mathcal{D}} \left[\left(Q(x_t, a_t) - r_t - \gamma \bar{V}(x_{t+1}) \right)^2 \right].$$

The target value function $\bar{V}$ is approximated via a Monte-Carlo estimate of the following expectation,

$$\bar{V}(x_{t+1}) = \mathbb{E}_{a_{t+1} \sim \pi} \left[\bar{Q}(x_{t+1}, a_{t+1}) - \alpha \log \pi(a_{t+1} | x_{t+1}) \right],$$

where $\bar{Q}$ is the target soft Q-function parameterized by a weight vector obtained from an exponentially moving average of the Q-function weights to stabilize training. The policy improvement step then attempts to project a parametric policy $\pi(a_t | x_t)$ by minimizing KL divergence between the policy and a Boltzmann distribution induced by the Q-function, producing the following objective,

$$J(\pi) = \mathbb{E}_{x_t \sim \mathcal{D}} \Big[\mathbb{E}_{a_t \sim \pi} [\alpha \log(\pi(a_t | x_t)) - Q(x_t, a_t)] \Big].$$

We provide the hyperparameters used for the RL experiments in table F.1.

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 F.1: A complete overview of used hyper parameters.

Contributions to joint-authored work

The research in this thesis is the product of a number of valuable collaborations.

Chapter 3 is based on work presented at the NeurIPS 2019 workshop on machine learning with guarantees [Lyle et al., 2019b]. All empirical results in this section were proposed and implemented by me. While I produced an initial analysis of PAC-Bayes bounds under symmetries and showed a preliminary version of Theorem 3.4.3 for finite groups independently, I worked closely with Benjamin Bloem-Reddy to generalize these initial results to the form presented in this thesis.

Chapter 4 is principally built on the NeurIPS 2020 paper [Lyle et al., 2020], but also draws on some of the insights developed in a tandem work led by Binxin Ru and myself which was presented at NeurIPS 2021 [Ru et al., 2021]. I was responsible for all theoretical and empirical results concerning linear models and gradient alignment, while Lisa Schut extended and improved on my initial toy experiments on neural networks to generate Figures 4.5 and B.4. Figure 4.6 was produced independently by me, but was inspired by analysis initially performed by Binxin Ru.

Chapter 5 was written in close collaboration with Mark Rowland and Will Dabney, and the initial research question sparking this project was brainstormed during joint meetings involving all authors. I ran the evaluations for Figure 5.3 and performed the evaluations on linear value function evolution in Appendix C.8. Mark proposed the continuous-time formalism and proved the initial subspace convergence result for a single value function, while I proposed the 'exact feature update' framework and proved the results concerning auxiliary tasks and ensemble prediction.

Chapter 6 is based on work I led during an internship at DeepMind [Lyle et al., 2022a]. I conducted additional analysis into the learned representations of the agents used for the empirical results of that work, and eventually formulated

the main hypotheses of this chapter based on that analysis. I wrote and ran the experiment code for the supervised and Atari benchmarks. My co-authors ran code I had written during my internship after I returned to Oxford to generate final versions of several figures, and provided helpful discussion throughout the project.

Chapter 7 is based on [Lyle et al., 2022b], for which I was the lead author. I was responsible for the implementations in all empirical results and for the proposal and initial proofs of the theoretical results. The other authors on the paper provided feedback on early drafts of the paper and recommendations for interesting experimental conditions to pursue.

Chapter 8 is based on a paper co-first-authored by myself and Amy Zhang. I stated and proved all theoretical results included in the thesis, proposed the algorithm for linear state abstractions, and implemented the linear model experiments. Both Amy and I worked together to propose the nonlinear MISA architecture and training objective.

Bibliography

Majid Abdolshah, Hung Le, Thommen Karimpanal George, Sunil Gupta, Santu Rana, and Svetha Venkatesh. A new representation of successor features for transfer across dissimilar environments. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 1–9. PMLR, 2021.

David Abel, Dilip Arumugam, Kavosh Asadi, Yuu Jinnai, Michael L Littman, and Lawson LS Wong. State abstraction as compression in apprenticeship learning (icml). In *Proceedings of the AAAI Conference on Artificial Intelligence. AAAI Press.*, 2019.

Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132: 428–446, 2020.

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. An optimistic perspective on offline reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Kartik Ahuja, Karthikeyan Shanmugam, Kush Varshney, and Amit Dhurandhar. Invariant risk minimization games. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 145–155. PMLR, 2020.

Hirotogu Akaike. *Information Theory and an Extension of the Maximum Likelihood Principle*, pages 199–213. Springer, 1998.

Francesco Alesiani, Shujian Yu, and Mathias Niepert. Continual invariant risk minimization, 2021.

Ron Amit and Ron Meir. Meta-learning by adjusting priors based on extended PAC-Bayes theory. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 205–214, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Oron Anschel, Nir Baram, and Nahum Shimkin. Averaged-DQN: Variance reduction and stabilization for deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

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

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 233–242. PMLR, 2017.

Jordan Ash and Ryan P Adams. On warm-starting neural network training. In *Advances in Neural Information Processing Systems*, 2020.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint*, 2016.

L Baird. Advantage updating. *Technical Report*, 1993.

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.

David Barrett and Benoit Dherin. Implicit gradient regularization. In *International Conference on Learning Representations (ICLR)*, 2021.

Peter L Bartlett. For valid generalization the size of the weights is more important than the size of the network. In *Advances in neural information processing systems*, pages 134–140, 1997.

Peter L Bartlett, Dylan J Foster, and Matus J Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017.

Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

Brian Bartoldson, Ari Morcos, Adrian Barbu, and Gordon Erlebacher. The generalization-stability tradeoff in neural network pruning. *Advances in Neural Information Processing Systems*, 33:20852–20864, 2020.

D. Basu. On statistics independent of a complete sufficient statistic. *Sankhyā: The Indian Journal of Statistics (1933-1960)*, 15(4):377–380, 1955. ISSN 00364452.

Eric B Baum and David Haussler. What size net gives valid generalization? In *Advances in neural information processing systems*, pages 81–90, 1989.

Nathan Beck, Durga Sivasubramanian, Apurva Dani, Ganesh Ramakrishnan, and Rishabh Iyer. Effective evaluation of deep active learning on image classification tasks. *arXiv preprint*, 2021.

Bahram Behzadian and Marek Petrik. Feature selection by singular value decomposition for reinforcement learning. In *ICML Prediction and Generative Modeling Workshop*, 2018.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine learning and the bias-variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Marc G Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, 2017.

Marc G Bellemare, Will Dabney, Robert Dadashi, Adrien Ali Taiga, Pablo Samuel Castro, Nicolas Le Roux, Dale Schuurmans, Tor Lattimore, and Clare Lyle. A geometric perspective on optimal representations for reinforcement learning. *Advances in neural information processing systems*, 2019.

Emmanuel Bengio, Valentin Thomas, Joelle Pineau, Doina Precup, and Yoshua Bengio. Independently controllable features. In *Reinforcement Learning and Decision Making (RLDM)*, 2017.

Emmanuel Bengio, Joelle Pineau, and Doina Precup. Interference and generalization in temporal difference learning (icml). In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 767–777. PMLR, 2020.

Yoshua Bengio, Mehdi Mirza, Ian Goodfellow, Aaron Courville, and Xia Da. An empirical investigation of catastrophic forgeting in gradient-based neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

Ari Benjamin, David Rolnick, and Konrad Kording. Measuring and regularizing networks in function space. In *International Conference on Learning Representations (ICLR)*, 2019.

Gregory W Benton, Wesley J Maddox, Sanae Lotfi, and Andrew Gordon Wilson. Loss surface simplexes for mode connecting volumes and fast ensembling. *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 18–24 Jul 2021.

Tudor Berariu, Wojciech Czarnecki, Soham De, Jorg Bornschein, Samuel Smith, Razvan Pascanu, and Claudia Clopath. A study on the plasticity of neural networks. *arXiv preprint arXiv:2106.00042*, 2021.

Dimitri P Bertsekas. Feature-based aggregation and deep reinforcement learning: A survey and some new implementations. *IEEE/CAA Journal of Automatica Sinica*, 6(1):1–31, 2018.

Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1st edition, 1996.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518): 859–877, 2017.

B. Bloem-Reddy and Y.W̃. Teh. Probabilistic symmetry and invariant neural networks. *Journal of Machine Learning Research*, 2019.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1613–1622, 2015.

Vivek S. Borkar and Sean P. Meyn. The o.d.e. method for convergence of stochastic approximation and reinforcement learning. *SIAM J. Control. Optim.*, 38:447–469, 2000.

Olivier Bousquet and André Elisseeff. Stability and generalization. *The Journal of Machine Learning Research*, 2:499–526, 2002.

Justin A Boyan. Least-squares temporal difference learning (icml). In *ICML*, pages 49–56. Citeseer, 1999.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

Sébastien Bubeck and Mark Sellke. A universal law of robustness via isoperimetry. *Advances in Neural Information Processing Systems*, 34, 2021.

Olivier Catoni. *PAC-Bayesian supervised classification: the thermodynamics of statistical Learning (ICML)*. Institute of Mathematical Statistics, 2007.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: Biasing gradient descent into wide valleys. *ICLR*, 2017.

Shuxiao Chen, Edgar Dobriban, and Jane H Lee. Invariance reduces variance: Understanding data augmentation in deep learning and beyond. *arXiv preprint arXiv:1907.10905*, 2019.

Wesley Chung, Somjit Nath, Ajin Joseph, and Martha White. Two-timescale networks for nonlinear value function approximation. In *International Conference on Learning Representations (ICLR)*, 2018.

Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 1282–1289. PMLR, 09–15 Jun 2019.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 2048–2056. PMLR, 2020.

Karl W Cobbe, Jacob Hilton, Oleg Klimov, and John Schulman. Phasic policy gradient. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 2020–2027. PMLR, 2021.

Taco S. Cohen and Max Welling. Group equivariant convolutional networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999. PMLR, 2016.

Taco S Cohen, Mario Geiger, and Maurice Weiler. A general theory of equivariant CNNs on homogeneous spaces. In *Advances in Neural Information Processing Systems 32*, 2019.

Ekin Dogus Cubuk, Barret Zoph, Dandelion Mané, Vijay Vasudevan, and Quoc V. Le. Autoaugment: Learning augmentation policies from data. *Computer Vision and Pattern Recognition (CVPR)*, 2019.

Wojciech M Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. Distilling policy distillation. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1331–1340. PMLR, 2019.

Will Dabney, Mark Rowland, Marc G Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *AAAI Conference on Artificial Intelligence*, 2018.

Will Dabney, André Barreto, Mark Rowland, Robert Dadashi, John Quan, Marc G Bellemare, and David Silver. The value-improvement path: Towards better

representations for reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, 2021.

Andreas Damianou and Neil Lawrence. Deep gaussian processes. In *Proceedings of Machine Learning Research*, volume 31, pages 207–215. PMLR, 2013.

Hadi Daneshmand, Amir Joudaki, and Francis Bach. Batch normalization orthogonalizes representations in deep random networks. In *Advances in Neural Information Processing Systems*, 2021.

Tri Dao, Albert Gu, Alexander J Ratner, Virginia Smith, Christopher De Sa, and Christopher Ré. A kernel theory of modern data augmentation. *Proceedings of the 36th International Conference on Machine Learning, PMLR 97*, 2019.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

Pim de Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *Advances in Neural Information Processing Systems*, 2019.

Giacomo De Palma, Bobak Kiani, and Seth Lloyd. Random deep neural networks are biased towards simple functions. *Advances in Neural Information Processing Systems*, 32, 2019.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, 2022.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1019–1028. JMLR. org, 2017.

Shibhansh Dohare, A Rupam Mahmood, and Richard S Sutton. Continual backprop: Stochastic gradient descent with persistent randomness. *arXiv preprint arXiv:2108.06325*, 2021.

Kefan Dong, Yuping Luo, Tianhe Yu, Chelsea Finn, and Tengyu Ma. On the expressivity of neural networks for deep reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 2627–2637. PMLR, 2020.

Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=HJxyZkBKDr.

Michele Donini, Luca Oneto, Shai Ben-David, John S Shawe-Taylor, and Massimiliano Pontil. Empirical risk minimization under fairness constraints. *Advances in Neural Information Processing Systems*, 31, 2018.

Simon S. Du, Akshay Krishnamurthy, Nan Jiang, Alekh Agarwal, Miroslav Dudík, and John Langford. Provably efficient RL with rich observations via latent state decoding. *International Conference on Machine Learning*, pages 1665–1674, 2019.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(61):2121–2159, 2011.

Vincent Dutordoir, Mark van der Wilk, Artem Artemev, and James Hensman. Bayesian image classification with deep convolutional gaussian processes. In *Proceedings of Machine Learning Research*, volume 108, pages 1529–1539. PMLR, 2020.

David Duvenaud, Dougal Maclaurin, and Ryan Adams. Early stopping as nonparametric variational inference. In *Artificial Intelligence and Statistics*, pages 1070–1077, 2016.

Gintare Karolina Dziugaite and Daniel M Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *UAI*, 2017.

Gintare Karolina Dziugaite and Daniel M Roy. Entropy-sgd optimizes the prior of a pac-bayes bound: Generalization properties of entropy-sgd and data-dependent priors. *Proceedings of the 35th International Conference on Machine Learning, PMLR 80*, pages :1377–1386, 2018a.

Gintare Karolina Dziugaite and Daniel M Roy. Data-dependent PAC-Bayes priors via differential privacy. In *Advances in Neural Information Processing Systems 31*, pages 8430–8441, 2018b.

Gintare Karolina Dziugaite, Alexandre Drouin, Brady Neal, Nitarshan Rajkumar, Ethan Caballero, Linbo Wang, Ioannis Mitliagkas, and Daniel M Roy. In search of robust measures of generalization. *Advances in Neural Information Processing Systems*, 33, 2020.

Frederick Eberhardt and Richard Scheines. Interventions and causal inference. *Philosophy of Science*, 74(5):981–995, 2007. doi: 10.1086/525638.

Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings, 2010.

Amir-massoud Farahmand. *Regularization in Reinforcement Learning*. PhD Thesis, University of Alberta, 2011.

Jesse Farebrother, Marlos C Machado, and Michael Bowling. Generalization and regularization in dqn. *arXiv preprint arXiv:1810.00123*, 2018.

Alhussein Fawzi, Horst Samulowitz, Deepak Turaga, and Pascal Frossard. Adaptive data augmentation for image classification. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3688–3692. Ieee, 2016.

William Fedus, Carles Gelada, Yoshua Bengio, Marc G Bellemare, and Hugo Larochelle. Hyperbolic discounting and learning over multiple horizons. In *Reinforcement Learning and Decision Making (RLDM)*, 2019a.

William Fedus, Dibya Ghosh, John D Martin, Marc G Bellemare, Yoshua Bengio, and Hugo Larochelle. On catastrophic interference in Atari 2600 games. *Reinforcement Learning and Decision Making*, 2019b.

Angelos Filos, Clare Lyle, Yarin Gal, Sergey Levine, Natasha Jaques, and Greg Farquhar. Psiphi-learning: Reinforcement learning with demonstrations using successor features and inverse temporal difference learning. *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 1126–1135. PMLR, 2017.

Stanislav Fort and Adam Scherlis. The goldilocks zone: Towards better understanding of neural network loss landscapes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3574–3581, 2019.

Stanislav Fort, Paweł Krzysztof Nowak, Stanislaw Jastrzebski, and Srini Narayanan. Stiffness: A new perspective on generalization in neural networks. *arXiv preprint arXiv:1901.09491*, 2019.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33, 2020.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Remi Munos, Demis Hassabis, et al. Noisy networks for exploration. In *International Conference on Learning Representations (ICLR)*, 2018.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 3259–3269. PMLR, 13–18 Jul 2020a.

Jonathan Frankle, David J. Schwab, and Ari S. Morcos. The early phase of neural network training. In *International Conference on Learning Representations (ICLR)*, 2020b.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1050–1059, 2016.

Carles Gelada, Saurabh Kumar, Jacob Buckman, Ofir Nachum, and Marc G. Bellemare. DeepMDP: Learning continuous latent space models for representation learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pages 2170–2179, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

Pascal Germain, Francis Bach, Alexandre Lacoste, and Simon Lacoste-Julien. Pac-bayesian theory meets bayesian inference. In *Advances in Neural Information Processing Systems*, pages 1884–1892, 2016a.

Pascal Germain, Francis Bach, Alexandre Lacoste, and Simon Lacoste-Julien. PAC-Bayesian theory meets Bayesian inference. In *Advances in Neural Information Processing Systems*, pages 1884–1892, 2016b.

Sina Ghiassian, Banafsheh Rafiee, Yat Long Lo, and Adam White. Improving performance in reinforcement learning by breaking generalization in neural networks. *arXiv preprint arXiv:2003.07417*, 2020.

Dibya Ghosh and Marc G Bellemare. Representations for stable off-policy reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

Florin Gogianu, Tudor Berariu, Mihaela Rosca, Claudia Clopath, Lucian Busoniu, and Razvan Pascanu. Spectral normalisation for deep reinforcement learning: an optimisation perspective. *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 3734–3744, 2021.

Aditya Golatkar, Alessandro Achille, and Stefano Soatto. Time matters in regularizing deep networks: Weight decay and data augmentation affect early learning dynamics, matter little near convergence. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, 2019.

Gene H Golub, Virginia C Klema, and Gilbert W Stewart. Rank degeneracy and least squares problems, 1976.

Anirudh Goyal, Riashat Islam, DJ Strouse, Zafarali Ahmed, Hugo Larochelle, Matthew Botvinick, Sergey Levine, and Yoshua Bengio. Transfer and exploration via the information bottleneck. In *International Conference on Learning Representations (ICLR)*, 2019.

Alex Graves. Practical variational inference for neural networks. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

Caglar Gulcehre, Srivatsan Srinivasan, Jakub Sygnowski, Georg Ostrovski, Mehrdad Farajtabar, Matt Hoffman, Razvan Pascanu, and Arnaud Doucet. An empirical study of implicit regularization in deep offline rl. *arXiv preprint arXiv:2207.02099*, 2022.

Alfred Haar. Der massbegriff in der theorie der kontinuierlichen gruppen. *Annals of mathematics*, pages 147–169, 1933.

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 (ICML)*, volume 80 of *Proceedings of Machine Learning Research*, pages 1861–1870, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Boris Hanin and David Rolnick. Deep relu networks have surprisingly few activation patterns. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Nicklas Hansen and Xiaolong Wang. Generalization in reinforcement learning by soft data augmentation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13611–13617. IEEE, 2021.

Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. PMLR, 2016.

Nick Harvey, Christopher Liaw, and Abbas Mehrabian. Nearly-tight VC-dimension bounds for piecewise linear neural networks. In Satyen Kale and Ohad Shamir, editors, *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 1064–1068. PMLR, 07–10 Jul 2017.

Bobby He, Balaji Lakshminarayanan, and Yee Whye Teh. Bayesian deep ensembles via the neural tangent kernel. *Advances in Neural Information Processing Systems*, 2020.

Hangfeng He and Weijie J Su. The local elasticity of neural networks. *International Conference on Learning Representations (ICLR)*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI conference on artificial intelligence*, 2018.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1): 1–42, 1997a.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997b.

Elad Hoffer, Itay Hubara, and Daniel Soudry. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *Advances in Neural Information Processing Systems*, pages 1731–1741, 2017.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

Peter J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964. ISSN 00034851.

Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. In *Advances in Neural Information Processing Systems*, 2019.

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 448–456. PMLR, 2015.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 633–644, October 2014.

P Izmailov, AG Wilson, D Podoprikhin, D Vetrov, and T Garipov. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 876–885, 2018.

Tommi Jaakkola, Michael I. Jordan, and Satinder P Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6), 1994.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations (ICLR)*, 2017.

Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized level replay. In *International Conference on Machine Learning*, pages 4940–4950. PMLR, 2021a.

Nan Jiang, Alex Kulesza, and Satinder Singh. Abstraction selection in model-based reinforcement learning. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.

Ray Jiang, Tom Zahavy, Zhongwen Xu, Adam White, Matteo Hessel, Charles Blundell, and Hado Van Hasselt. Emphatic algorithms for deep reinforcement learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pages 5023–5033. PMLR, 18–24 Jul 2021b.

Yiding Jiang, Behnam Neyshabur, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations (ICLR)*, 2020.

Chi Jin, Zhuoran Yang, Zhaoran Wang, and Michael I Jordan. Provably efficient reinforcement learning with linear function approximation. In *Conference on Learning Theory*, pages 2137–2143. PMLR, 2020.

Fredrik Johansson, Uri Shalit, and David Sontag. Learning representations for counterfactual inference. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 3020–3029, 2016.

Nicholas K Jong and Peter Stone. State abstraction discovery from irrelevant state variables. In *IJCAI*, volume 8, pages 752–757. Citeseer, 2005.

Dimitris Kalimeris, Gal Kaplun, Preetum Nakkiran, Benjamin Edelman, Tristan Yang, Boaz Barak, and Haofeng Zhang. Sgd on neural networks learns functions of increasing complexity. *Advances in Neural Information Processing Systems*, 32:3496–3506, 2019.

Pritish Kamath, Akilesh Tangella, Danica Sutherland, and Nathan Srebro. Does invariant risk minimization capture invariance? In Arindam Banerjee and Kenji Fukumizu, editors, *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, volume 130 of *Proceedings of Machine Learning Research*, pages 4069–4077. PMLR, 13–15 Apr 2021.

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. *ICLR*, 2017.

Mohammad Emtiyaz E Khan, Alexander Immer, Ehsan Abedi, and Maciej Korzepa. Approximate inference turns deep networks into gaussian processes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3094–3104. Curran Associates, Inc., 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. A survey of generalisation in deep reinforcement learning. *arXiv preprint arXiv:2111.09794*, 2021.

James Kirkpatrick, Razvan Pascanu, Neil 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. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017.

Frederic Koehler, Lijia Zhou, Danica Sutherland, and Nathan Srebro. Uniform convergence of interpolators: Gaussian width, norm bounds and benign overfitting. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 20657–20668. Curran Associates, Inc., 2021.

Vijay R Konda and John N Tsitsiklis. Actor-critic algorithms. In *Advances in neural information processing systems*, pages 1008–1014, 2000.

Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *Proc. ICML 35*, volume 80 of *PMLR*, pages 2747–2755, 2018.

Jannik Kossen, Neil Band, Clare Lyle, Aidan N Gomez, Tom Rainforth, and Yarin Gal. Self-attention between datapoints: Going beyond individual input-output pairs in deep learning. *Advances in Neural Information Processing Systems*, 2021.

Anders Krogh and John Hertz. A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4, 1991.

S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22(1):79–86, 03 1951. doi: 10.1214/aoms/1177729694.

Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2021.

Aviral Kumar, Rishabh Agarwal, Tengyu Ma, Aaron Courville, George Tucker, and Sergey Levine. Dr3: Value-based deep reinforcement learning requires explicit regularization. *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022.

Heinrich Küttler, Nantas Nardelli, Alexander Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. The nethack learning environment. *Advances in Neural Information Processing Systems*, 33:7671–7684, 2020.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017.

Charline Le Lan, Stephen Tu, Adam Oberman, Rishabh Agarwal, and Marc G Bellemare. On the generalization of representations in reinforcement learning. *Artificial Intelligence and Statistics (AISTATS)*, 2022.

John Langford and John Shawe-Taylor. Pac-bayes & margins. *Advances in neural information processing systems*, pages 439–446, 2003.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. CURL: Contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020a.

Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in Neural Information Processing Systems*, 33:19884–19895, 2020b.

Jaehoon Lee, Jascha Sohl-dickstein, Jeffrey Pennington, Roman Novak, Sam Schoenholz, and Yasaman Bahri. Deep neural networks as gaussian processes. In *International Conference on Learning Representations (ICLR)*, 2018.

Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Advances in neural information processing systems*, 32, 2019.

Jaehoon Lee, Samuel Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. *Advances in Neural Information Processing Systems*, 33, 2020.

Kimin Lee, Laura M Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 6152–6163. PMLR, 2021.

Guy Lever, François Laviolette, and John Shawe-Taylor. Tighter PAC-Bayes bounds through distribution-dependent priors. *Theoretical Computer Science*, 473:4–28, 2013.

Nir Levine, Tom Zahavy, Daniel J Mankowitz, Aviv Tamar, and Shie Mannor. Shallow updates for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2017.

Alex Lewandowski. Generalization across space and time in reinforcement learning. *NeurIPS Pre-Registration in Machine Learning Workshop*, 2020.

Alexander Li and Deepak Pathak. Functional regularization for reinforcement learning via learned fourier features. *Advances in Neural Information Processing Systems*, 34, 2021.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 6391–6401, 2018.

Lihong Li, Thomas J Walsh, and Michael L Littman. Towards a unified theory of state abstraction for mdps. In *ISAIM*, 2006.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017.

Xingyu Lin, Harjatin Baweja, George Kantor, and David Held. Adaptive auxiliary task weighting for reinforcement learning. In *Advances in Neural Information Processing Systems*, 2019.

Vincent Liu, Adam White, Hengshuai Yao, and Martha White. Towards a practical measure of interference for reinforcement learning. *arXiv preprint arXiv:2007.03807*, 2020a.

Vincent Liu, Adam M White, Hengshuai Yao, and Martha White. Measuring and mitigating interference in reinforcement learning. *arXiv preprint*, 2020b.

Yat Long Lo and Sina Ghiassian. Overcoming catastrophic interference in online reinforcement learning with dynamic self-organizing maps. *arXiv preprint arXiv:1910.13213*, 2019.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning (icml). In *Advances in Neural Information Processing Systems*, 2017.

Christos Louizos, Uri Shalit, Joris M Mooij, David Sontag, Richard Zemel, and Max Welling. Causal effect inference with deep latent-variable models. In *Advances in Neural Information Processing Systems*, pages 6446–6456, 2017.

Clare Lyle, Marc G Bellemare, and Pablo Samuel Castro. A comparative analysis of expected and distributional reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2019a.

Clare Lyle, Mark van der Wilk, Marta Kwiatkowska, Yarin Gal, and Benjamin Bloem-Reddy. On the benefits of invariance in neural networks. *NeurIPS Workshop on Machine Learning with Guarantees*, 2019b.

Clare Lyle, Lisa Schut, Robin Ru, Yarin Gal, and Mark van der Wilk. A bayesian perspective on training speed and model selection. *Advances in Neural Information Processing Systems*, 33, 2020.

Clare Lyle, Mark Rowland, Georg Ostrovski, and Will Dabney. On the effect of auxiliary tasks on representation dynamics. In *Artificial Intelligence and Statistics (AISTATS)*, 2021a.

Clare Lyle, Amy Zhang, Minqi Jiang, Joelle Pineau, and Yarin Gal. Resolving causal confusion in reinforcement learning via robust exploration. In *Self-Supervision for Reinforcement Learning Workshop - ICLR 2021*, 2021b.

Clare Lyle, Mark Rowland, and Will Dabney. Understanding and preventing capacity loss in reinforcement learning. In *International Conference on Learning Representations (Spotlight)*, 2022a.

Clare Lyle, Mark Rowland, Will Dabney, Marta Kwiatkowska, and Yarin Gal. Learning dynamics and generalization in reinforcement learning. *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022b.

Marlos C. Machado, Marc G. Bellemare, and Michael Bowling. A Laplacian framework for option discovery in reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pages 2295–2304. PMLR, 06–11 Aug 2017.

Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the Arcade Learning Environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018a.

Marlos C Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. In *International Conference on Learning Representations (ICLR)*, 2018b.

David J. C. MacKay. Choice of basis for laplace approximation. *Journal of Machine Learning*, 33(1), 1998.

David JC MacKay. *Bayesian methods for adaptive models*. PhD thesis, California Institute of Technology, 1992.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *Advances in Neural Information Processing Systems*, pages 13132–13143, 2019.

Wesley J Maddox, Gregory Benton, and Andrew Gordon Wilson. Rethinking parameter counting in deep models: Effective dimensionality revisited. *arXiv preprint arXiv:2003.02139*, 2020.

Sara Magliacane, Thijs van Ommen, Tom Claassen, Stephan Bongers, Philip Versteeg, and Joris M Mooij. Domain adaptation by using causal inference to predict invariant conditional distributions. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 10846–10856. Curran Associates, Inc., 2018.

Sridhar Mahadevan. Proto-value functions: Developmental reinforcement learning. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 553–560, 2005.

Sridhar Mahadevan. Learning representation and control in Markov decision processes: New frontiers. *Foundations and Trends® in Machine Learning (ICML)*, 1(4):403–565, 2009.

Sridhar Mahadevan and Mauro Maggioni. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *Journal of Machine Learning Research*, 8(Oct):2169–2231, 2007.

Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *The Journal of Machine Learning Research*, 18(1):4873–4907, 2017.

Alexander G de G Matthews, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Sample-then-optimize posterior sampling for bayesian linear models. *Advances in Neural Information Processing Systems*, 2017.

Alexander G. de G. Matthews, Jiri Hron, Mark Rowland, Richard E. Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

David A. McAllester. Some PAC-Bayesian Theorems. *Machine Learning (ICML)*, 37(3):355–363, 1999.

Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. Learning to navigate in complex environments. In *International Conference on Learning Representations (ICLR)*, 2017.

Dipendra Misra, Mikael Henaff, Akshay Krishnamurthy, and John Langford. Kinematic state abstraction and provably efficient rich-observation reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 6961–6971. PMLR, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529–533, 2015.

Ari S. Morcos, David G.T. Barrett, Neil C. Rabinowitz, and Matthew Botvinick. On the importance of single directions for generalization. In *International Conference on Learning Representations (ICLR)*, 2018.

Vaishnavh Nagarajan and J. Zico Kolter. Uniform convergence may be unable to explain generalization in deep learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d' Alche-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11615–11626. Curran Associates, Inc., 2019.

Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Jeffrey Negrea, Gintare Karolina Dziugaite, and Daniel Roy. In defense of uniform convergence: Generalization via derandomization with an application to interpolating predictors. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 7263–7272. PMLR, 2020.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. *International Conference on Learning Representations (Workshop Track)*, 2015a.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning (ICML) Theory*, pages 1376–1401, 2015b.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956, 2017.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. The role of over-parametrization in generalization of neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Evgenii Nikishin, Max Schwarzer, Pierluca D'Oro, Pierre-Luc Bacon, and Aaron Courville. The primacy bias in deep reinforcement learning. In *Proceedings of the 39th International Conference on International Conference on Machine Learning (ICML)*, 2022.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In *Advances in Neural Information Processing Systems*, 2016.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 8617–8629, 2018.

Georg Ostrovski, Pablo Samuel Castro, and Will Dabney. The difficulty of passive learning in deep reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *arXiv preprint*, 2018.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob Foerster, Edward Grefenstette, and Tim Rocktäschel. Evolving curricula with regret-based environment design. *arXiv preprint arXiv:2203.01302*, 2022.

Ronald Parr, Lihong Li, Gavin Taylor, Christopher Painter-Wakefield, and Michael L Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008.

Judea Pearl. *Causality: models, reasoning and inference*, volume 29. Springer, 2000.

Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):947–1012, 2016.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. MIT press, 2017.

Tobias Pohlen, Bilal Piot, Todd Hester, Mohammad Gheshlaghi Azar, Dan Horgan, David Budden, Gabriel Barth-Maron, Hado Van Hasselt, John Quan, Mel Večerík, et al. Observe and look further: Achieving consistent performance on atari. *arXiv preprint arXiv:1805.11593*, 2018.

Karl R Popper. *The logic of scientific discovery, Karl R. Popper*. Hutchinson. London. GB, 1968.

Doina Precup, Richard S Sutton, and Sanjoy Dasgupta. Off-policy temporal-difference learning with function approximation. In *ICML*, pages 417–424, 2001.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994. ISBN 978-0-47161977-2. doi: 10.1002/9780470316887.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, pages 652–660, 2017.

John Quan and Georg Ostrovski. DQN Zoo: Reference implementations of DQN-based agents, 2020.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *arXiv preprint*, 2018.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl-Dickstein. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 2847–2854. PMLR, 2017.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 5301–5310. PMLR, 2019.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. *International Conference on Machine Learning (ICML)*, pages 8787–8798, 2021.

Roberta Raileanu, Maxwell Goldstein, Denis Yarats, Ilya Kostrikov, and Rob Fergus. Automatic data augmentation for generalization in reinforcement learning. In *Advances in Neural Information Processing Systems*, 2021.

Anant Raj, Abhishek Kumar, Youssef Mroueh, Tom Fletcher, and Bernhard Schoelkopf. Local Group Invariant Representations via Orbit Embeddings. In Aarti Singh and Jerry Zhu, editors, *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pages 1225–1235. PMLR, 2017.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 5331–5340. PMLR, 2019.

Carl Edward Rasmussen. Gaussian processes in machine learning (icml). In *Summer School on Machine Learning (ICML)*, pages 63–71. Springer, 2003.

Carl Edward Rasmussen and Zoubin Ghahramani. Occam's razor. In *Advances in neural information processing systems*, pages 294–300, 2001.

Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Equivariance through parameter-sharing. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2892–2901. JMLR. org, 2017.

Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

Mateo Rojas-Carulla, Bernhard Schölkopf, Richard Turner, and Jonas Peters. Invariant models for causal transfer learning. *The Journal of Machine Learning Research*, 19(1):1309–1342, 2018.

Binxin Ru, Clare Lyle, Lisa Schut, Mark van der Wilk, and Yarin Gal. Revisiting the train loss: an efficient performance estimator for neural architecture search. *Advances in Neural Information Processing Systems (Spotlight Presentation)*, 34, 2021.

Andrei A Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. In *ICLR (Poster)*, 2016.

Justin Salamon and Juan Pablo Bello. Deep convolutional neural networks and data augmentation for environmental sound classification. *IEEE Signal Processing Letters*, 24(3):279–283, 2017.

Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Kuttler, Edward Grefenstette, and Tim Rocktäschel. Minihack the planet: A sandbox for open-ended reinforcement learning research. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.

Tom Schaul, Diana Borsa, Joseph Modayil, and Razvan Pascanu. Ray interference: a source of plateaus in deep reinforcement learning. *Workshop on Reinforcement Learning and Decision Making*, 2019.

Simon Schmitt, Jonathan J Hudson, Augustin Zidek, Simon Osindero, Carl Doersch, Wojciech M Czarnecki, Joel Z Leibo, Heinrich Kuttler, Andrew Zisserman, Karen Simonyan, et al. Kickstarting deep reinforcement learning. *arXiv preprint arXiv:1803.03835*, 2018.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning (icml). In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 2018.

Jean-Pierre Serre. *Linear representations of finite groups*, volume 42. Springer, 1977.

Uri Shalit, Fredrik D Johansson, and David Sontag. Estimating individual treatment effect: generalization bounds and algorithms. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3076–3085. JMLR. org, 2017.

Lin Shao, Yifan You, Mengyuan Yan, Qingyun Sun, and Jeannette Bohg. GRAC: self-guided and self-regularized actor-critic. *Conference on Robot Learning*, pages 267–276, 2022.

Noel E Sharkey and Amanda JC Sharkey. An analysis of catastrophic interference. *Connection Science*, 1995.

Daniel L Silver and Robert E Mercer. The task rehearsal method of life-long learning: Overcoming impoverished data. In *Conference of the Canadian Society for Computational Studies of Intelligence*, pages 90–101, 2002.

Samuel L. Smith and Quoc V. Le. A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations (ICLR)*, 2018.

Samuel L Smith, Benoit Dherin, David Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations (ICLR)*, 2020.

Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based tepresentations. In *International Conference on Machine Learning (ICML)*, pages 9767–9779. PMLR, 2021.

Xingyou Song, Yiding Jiang, Stephen Tu, Yilun Du, and Behnam Neyshabur. Observational overfitting in reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Kimberly L Stachenfeld, Matthew Botvinick, and Samuel J Gershman. Design principles of the hippocampal cognitive map. In *Advances in Neural Information Processing Systems*, 2014.

Kimberly L Stachenfeld, Matthew M Botvinick, and Samuel J Gershman. The hippocampus as a predictive map. *Nature neuroscience*, 20(11):1643–1653, 2017.

Raphael Suter, Djordje Miladinovic, Bernhard Schölkopf, and Stefan Bauer. Robustly disentangled causal mechanisms: Validating deep representations for interventional robustness. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 6056–6065. PMLR, 2019.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

Richard S Sutton, Csaba Szepesvári, and Hamid Reza Maei. A convergent $O(n)$ algorithm for off-policy temporal-difference learning with linear function approximation. *Advances in Neural Information Processing Systems*, 2008.

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.

Jonathan Taylor, Doina Precup, and Prakash Panagaden. Bounding performance loss in approximate mdp homomorphisms. *Advances in Neural Information Processing Systems*, 21, 2008.

Yee Teh, Victor Bapst, Wojciech M. Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4496–4506. Curran Associates, Inc., 2017.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *2015 ieee information theory workshop (itw)*, pages 1–5. IEEE, 2015.

John Tsitsiklis and Benjamin Van Roy. Analysis of temporal-diffference learning with function approximation. *Advances in neural information processing systems*, 9, 1996.

John N Tsitsiklis. Asynchronous stochastic approximation and Q-learning (icml). *Machine Learning (ICML)*, 16(3):185–202, 1994.

E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2962–2971, Los Alamitos, CA, USA, jul 2017. IEEE Computer Society. doi: 10.1109/CVPR.2017.316.

Karen Ullrich, Edward Meeds, and Max Welling. Soft weight-sharing for neural network compression. In *International Conference on Learning Representations (ICLR)*, 2017.

Guillermo Valle-Pérez, Chico Q Camargo, and Ard A Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. *International Conference on Learning Representations*, 2018.

Elise van der Pol, Thomas Kipf, Frans A Oliehoek, and Max Welling. Plannable approximations to mdp homomorphisms: Equivariance under actions. *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1431–1439, 2020a.

Elise van der Pol, Daniel Worrall, Herke van Hoof, Frans Oliehoek, and Max Welling. Mdp homomorphic networks: Group symmetries in reinforcement learning. *Advances in Neural Information Processing Systems*, 33:4199–4210, 2020b.

Mark van der Wilk, Matthias Bauer, ST John, and James Hensman. Learning invariances using the marginal likelihood. In *Advances in Neural Information Processing Systems*, pages 9938–9948, 2018.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning (icml). In *AAAI Conference on Artificial Intelligence*, 2016.

Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

Vladimir Vapnik. On the uniform convergence of frequencies of occurrence of events to their probabilities. *Theory of Probability and its Applications*, 16, 1968.

Vladimir Vapnik. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4, 1991.

Vladimir Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1999.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Vivek Veeriah, Matteo Hessel, Zhongwen Xu, Richard Lewis, Janarthanan Rajendran, Junhyuk Oh, Hado van Hasselt, David Silver, and Satinder Singh. Discovery of useful questions as auxiliary tasks. In *Advances in Neural Information Processing Systems*, 2019.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas,

Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

B Wang, C Lyle, and M Kwiatkowska. Provable guarantees on the robustness of decision rules to causal interventions. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2021.

Yining Wang, Ruosong Wang, Simon Shaolei Du, and Akshay Krishnamurthy. Optimism in reinforcement learning with generalized linear function approximation. In *International Conference on Learning Representations (ICLR)*, 2020.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine Learning (ICML)*, 8(3-4):279–292, 1992.

Colin Wei and Tengyu Ma. Improved sample complexities for deep neural networks and robust classification via an all-layer margin. In *International Conference on Learning Representations (ICLR)*, 2019.

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, 2011.

Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *Advances in neural information processing systems*, 33:4697–4708, 2020.

Jeffrey Wood and John Shawe-Taylor. Representation theory and invariant neural networks. *Discrete Applied Mathematics*, 69(1):33–60, 1996.

Sen Wu, Hongyang Zhang, Gregory Valiant, and Christopher Re. On the generalization effects of linear transformations in data augmentation. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference*

on Machine Learning (ICML), volume 119 of *Proceedings of Machine Learning Research*, pages 10410–10420. PMLR, 13–18 Jul 2020.

Ge Yang and Samuel Schoenholz. Mean field residual networks: On the edge of chaos. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Ge Yang, Anurag Ajay, and Pulkit Agrawal. Overcoming the spectral bias of neural value approximation. In *International Conference on Learning Representations (ICLR)*, 2022.

Greg Yang and Edward J Hu. Tensor programs iv: Feature learning in infinite-width neural networks. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, pages 11727–11737. PMLR, 2021.

Denis Yarats and Ilya Kostrikov. Soft actor-critic (sac) implementation in pytorch. https://github.com/denisyarats/pytorch_sac, 2020.

Ke Ye and Lek-Heng Lim. Schubert varieties and distances between subspaces of different dimensions. *SIAM Journal on Matrix Analysis and Applications*, 37(3): 1176–1197, 2016.

Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. *International Conference on Learning Representations (ICLR)*, 2019.

Sheheryar Zaidi, Tudor Berariu, Hyunjik Kim, Jörg Bornschein, Claudia Clopath, Yee Whye Teh, and Razvan Pascanu. When does re-initialization work? *arXiv preprint arXiv:2206.10011*, 2022.

Amy Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *arXiv preprint*, 2018a.

Amy Zhang, Yuxin Wu, and Joelle Pineau. Natural environment benchmarks for reinforcement learning. *arXiv preprint*, abs/1811.06032, 2018b.

Amy Zhang, Zachary C. Lipton, Luis Pineda, Kamyar Azizzadenesheli, Anima Anandkumar, Laurent Itti, Joelle Pineau, and Tommaso Furlanello. Learning causal state representations of partially observable environments. *Reinforcement Learning and Decision Making*, 2019.

Amy Zhang, Clare Lyle, Shagun Sodhani, Angelos Filos, Marta Kwiatkowska, Joelle Pineau, Yarin Gal, and Doina Precup. Invariant causal prediction for block MDPs. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020a.

Amy Zhang, Rowan Thomas McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. In *International Conference on Learning Representations (ICLR)*, 2020b.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017.

Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint*, 2018c.

R. Zhao, D. Wang, R. Yan, K. Mao, F. Shen, and J. Wang. Machine health monitoring using local feature-based gated recurrent unit networks. *IEEE Transactions on Industrial Electronics*, 65(2):1539–1548, Feb 2018.

Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning-based sequence model. *Nature Methods*, 12(10):931, 2015.

Wenda Zhou, Victor Veitch, Morgane Austern, Ryan P. Adams, and Peter Orbanz. Non-vacuous generalization bounds at the imagenet scale: a PAC-Bayesian

compression approach. *International Conference on Learning Representations (ICLR)*, 2019.

Julian G. Zilly, Alessandro Achille, Andrea Censi, and Emilio Frazzoli. On plasticity, invariance, and mutually frozen weights in sequential task learning (icml). In *Advances in Neural Information Processing Systems*, 2021.

Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning (icml). In *International Conference on Machine Learning (ICML)*, pages 7693–7702. PMLR, 2019.

Will Zou, Shenghuo Zhu, Kai Yu, and Andrew Y Ng. Deep learning of invariant features via simulated fixations in video. In *Advances in Neural Information Processing Systems*, pages 3203–3211, 2012.