Learning to model what matters – Representations and world models for efficient reinforcement learning

by

Claas Alexander Voelcker

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy

Department of Computer Science
University of Toronto

Learning to model what matters – Representations and world models for efficient reinforcement learning

Claas Alexander Voelcker
Doctor of Philosophy

Department of Computer Science
University of Toronto
2025

Abstract

To write

Acknowledgements

Contents

Preface

As with all works of science, this work stands on the shoulders of giants and has been supported generously by my collaborators and advisors. Throughout the thesis, I make references to sources for know results following the standards of scientific research.

In this thesis, I summarize and present the results of my research conducted at the University of Toronto. As Machine Learning has become a collaborative discipline, much of my own work has been developed together with my advisors, colleagues at the University and abroad, and the amazing students who I had the great fortune to mentor. This thesis is built on work developed in collaboration with Amir-massoud Farahmand, Igor Gilitschenski, Animesh Garg, Eric Eaton, Marcel Hussing, Romina Abachi, Arash Ahmadian, Victor Liao, and Anastasiia Pedan. I acknowledge and thank them for their great work.

This section clarifies the individual contributions in each chapter. With all papers, all authors were involved in editing the writing. The chapters in this thesis are lightly edited from the original papers to provide clearer insights in the context of the overarching narrative.

- Chapter 3: This chapter is based on Marcel Hussing, Claas A Voelcker, Igor Gilitschenski, Amir-massoud Farahmand, and Eric Eaton (2024). "Dissecting Deep RL with High Update Ratios: Combatting Value Divergence". In: *Reinforcement Learning Conference*. I contributed as joint first author and the core ideas of the work were developed in collaboration. Marcel Hussing provided the initial experiments on priming, the effects of optimizers, regularizers and conservative learning. The experiments on feature normalization were designed jointly. Writing was shared primarily between Marcel Hussign an

- Chapter 4: This chapter is based on Claas A Voelcker, Victor Liao, Animesh Garg, and Amir-massoud Farahmand (2022). "Value Gradient weighted Model-Based Reinforcement Learning". In: *International Conference on Learning Representations*. The writing, insights, core methodology, and experiments are my contribution, Victor Liao contributed to the experimental infrastructure and code.

- Chapter 5: This chapter is based on Claas A Voelcker, Tyler Kastner, Igor Gilitschenski, and Amir-massoud Farahmand (2024). "When does self-prediction help? Understanding Auxiliary Tasks in Reinforcement Learning". In: *Reinforcement Learning Conference*. The writing, insights, proofs, and experiments are my contribution. The proofs in **??** are based on joint work with Tyler Kastner. **??** was suggested by Amin Rakhsha and fully written out by me.

- Chapter 6: This chapter is based on Claas A Voelcker, Anastasiia Pedan, Arash Ahmadian, Romina Abachi, Igor Gilitschenski, and Amir-massoud Farahmand (2025). "Callibrated value-aware loss functions with stochastic environment models". In: *under review*, and an older version of the paper publicized as Claas A Voelcker, Arash Ahmadian, Romina Abachi, Igor Gilitschenski, and Amir-massoud Farahmand (2024). *λ-models: Effective Decision-Aware Reinforcement Learning with Latent Models*. arXiv: 2306.17366 [cs.LG]. URL: https://arxiv.org/abs/2306.17366. The insights and most proofs are my contribution. The experimental architecture was first developed in colaboration with Romina Abachi, and Arash Ahmadian, and greatly expanded by Anastasiia Pedan. The final experiments were designed by me. **??** was conjectured by me, the proof was thoroughly sketched out by Amir-massoud Farahmand, and finally written out and completed by me. **??** was proposed by Amin Rakhsha and fully written out by me.

- Chapter 7: This chapter is based on Claas A Voelcker, Marcel Hussing, Eric Eaton, Amir-massoud Farahmand, and Igor Gilitschenski (2025). "MAD-TD: Model-Augmented Data stabilizes High Update Ratio RL". in: *International Conference on Learning Representations*. I contributed as joint first author and the core ideas of the work were developed in collaboration. The initial insights of the paper were developed based on previous joint work Hussing et al. 2024 together with Marcel Hussing as well as my own work in Voelcker, Ahmadian, et al. (2024). The initial experiments on the impact of model data on on- and off-policy validation loss were contributed by me. All further experiments, the core architecture and experimental setup were developed jointly.

In addition to the papers listed above, I contributed to the following reinforcement learning works during my time at the University of Toronto. These papers are not primary chapters in this thesis, and where their ideas are used, they are cited accordingly

- Romina Abachi, Claas A Voelcker, Animesh Garg, and Amir-massoud Farahmand (2022). "VIPer: Iterative Value-Aware Model Learning on the Value Improvement Path". In: *Decision Awareness in Reinforcement Learning Workshop at ICML 2022*: The paper presents a latent model-based architecture that uses an IterVAML (Farahmand 2018) style loss over past value function estimates. The architecture was first

implemented in Voelcker, Ahmadian, et al. (2024), Voelcker, Hussing, Eaton, et al. (2025), and Voelcker, Pedan, et al. (2025) and the past value functions were discarded in favor of other, more computationally favorable auxiliary losses investigated in Voelcker, Kastner, et al. (2024). I contributed to the idea and writing of the paper.

- Jonas Guan, Shon Eduard Verch, Claas A Voelcker, Ethan C Jackson, Nicolas Papernot, and William A Cunningham (2024). "Temporal-Difference Learning Using Distributed Error Signals". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. URL: https://openreview.net/forum?id=8moTQjfqAV: The paper presents a novel architecture for training reinforcement learning agents without backpropagating TD learning signals through a neural architecture. As backpropagation is not a biologically plausible learning architecture, this work provides an important proof-of-feasibiity for non-backpropagation based reinforcement learning. I served as a reinforcement learning expert on the paper and helped developed the experiments.

- Claas A Voelcker, Marcel Hussing, and Eric Eaton (2024). "Can we hop in general? A discussion of benchmark selection and design using the Hopper environment". In: *Finding the Frame: An RLC Workshop for Examining Conceptual Frameworks*. URL: https://openreview.net/forum?id=9IgtF63LPA: This paper presents insights into the impact of environment selection on the empirical evaluation of reinforcement learning algorithms. The core ideas of the paper were developed jointly with Marcel Hussing and experimental evaluation was done jointly. The insights informed environment selection for Voelcker, Hussing, Eaton, et al. (2025).

Chapter 1

Introduction

MORE CITATIONS

Agents that act autonomously and intelligently in a complex world are in many ways the ultimate goal of AI research. An industrial robot in a highly controlled environment, such as a car assembly line, can be programmed to behave reliably and safely by human engineers. However, if we want robots to assist us in the real world outside of such controlled circumstances, we need to endow them with the ability to test out new behaviors and learn from their surroundings. This goal has been pursued most comprehensively under the banner of *reinforcement learning*, in which an agent learns by sequentially interacting with an environment and receiving a scalar reward.

When building algorithms to act in unknown environments, one core challenge is to model the surroundings of the agent efficiently. The industrial robot can be tightly controlled and we can pre-specify what it will encounter in its limited environment and how to react to it. But an autonomous agent needs to be able to perceive all of its surroundings and decide what information is necessary for its task. In order to make decisions, such agents therefore need to filter and process their sensory input and learn about the structure of the environment, the consequences of their actions, and the rewards associated with different actions.

In supervised learning, neural networks have shown a remarkable ability to extract relevant information automatically from raw data, such as images or text, to solve complex tasks (Goodfellow et al. 2016). However, in reinforcement learning, where an agent needs to learn how to act in an environment through trial and error, this task has proven to be more difficult. As the agent explores the environment gradually, it is not easy to distinguish from the start what information is relevant for the decision making task and what is not. Failing to do so often leads to agents getting stuck in suboptimal policies, overfitting to irrelevant aspects of the environment, or failing to learn anything as they lack the computational

capacity to model the world. Hence, reinforcement learning algorithms do not yet fulfill their promise of efficient and reliable learning in wide, complex environments, let alone the real world.

1.1 Problem statement

In this thesis we will investigate how to build algorithms which can learn to model complex environments. Our main focus will be on efficient and stable representation and world model learning for value estimation. Representations and world models are two important tools that allow an agent to process information. A representation is a function which takes the sensory input of an agent, such as a camera feed or low-level joint information of a robot, and produces a numerical representation that is useful for reinforcement learning (Ferns, Panangaden, and Precup 2004; Jaderberg et al. 2017; Abel 2020; Le Lan, Marc G Bellemare, and Castro 2021). A world model allows an agent to predict how its actions impact the environment, and therefore it can be used to improve value estimation or for planning (Richard S Sutton 1990; Janner et al. 2019a; Hafner et al. 2020; Schrittwieser et al. 2020). As we will see, world model learning and representation learning are closely related concepts which can be used jointly for stable and efficient value learning.

1.1.1 The problem of unstable value function learning

To understand why representation learning and world models can help us, we first need to understand the problems (deep) reinforcement learning faces. In this thesis, we will focus primarily on the stability of value estimation. A value function is an important concept in reinforcement learning. It summarizes the expected future return of executing a policy in a state. Value functions can be used to judge the impact of taking an action, to compare and improve policies, and to guide exploration. Given their broad usefulness, a lot of effort has been devoted to obtain good value function learning algorithms, especially using deep neural networks (Mnih et al. 2013; H. v. Hasselt 2010; H. v. Hasselt, Guez, and Silver 2016; Fujimoto, Hoof, and Meger 2018; Haarnoja et al. 2018a). However, stable and efficient value learning in online scenarios with function approximation remains an important unsolved problem, and many recent works show persistent problems with existing learning algorithms (Kumar et al. 2021; Nikishin, Schwarzer, et al. 2022; Hussing et al. 2024).

As discussed, one central issue with learning value functions stems from the problem of reliably extracting useful information from interaction data. While neural networks can extract useful features automatically in supervised learning setups, but fail to do so in reinforcement learning. To improve this, we will investigate how to combine *decision-aware* algorithms, approaches that learn tailored representations and world models for the decision-making task and agent is attempting, with *general-purpose* algorithms, approaches

that simply attempt to generate representations and models which can be used for any downstream task.

More concretely, *decision-awareness* describes the idea that the components of a learning approach for decision making, such as representations and models, should account for the final decision task (Farahmand, André Barreto, and Nikovski 2017; Grimm, André Barreto, et al. 2020; Abachi, Ghavamzadeh, and Farahmand 2020; Nikishin, Abachi, et al. 2022). In many cases, these components can also be trained in a <emphgeneral-purpose way using objectives such as maximum likelihood estimation However this can be inefficient or suboptimal, as computational resources are wasted on learning irrelevant aspects of the decision-making task. Instead, a decision-aware algorithms will account for the decision making task and e.g. put more emphasis on certain parts of the observation space, such as a traffic light, instead of others, such as clouds moving in the background of a video feed.

While this suggests that we should always use decision-aware approaches, experimental evidence complicates this. Even though decision-aware algorithms promise to be more resource efficient, training with general-purpose methods, for example fitting a world model with a maximum likelihood objective, turns out to be more stable in practice (Voelcker, Liao, et al. 2022). Decision-aware algorithms need to continually adapt to the decision task as more information becomes available, while general purpose methods only have a single, stable training objective throughout the learning process. Especially in the beginning of training when available information about the decision task is limited, this can lead to instability or suboptimal convergence of the learning algorithm.

To address these challenges, I put forward the following thesis.

> To solve the persistent issues of *instability* and *inefficiency* of value function learning and to successfully leverage *decision-aware* learning, an agent needs to effectively combine available information on the decision task with general purpose learning mechanisms.

Achieving this goal requires us to understand how we can build strong algorithms that use the best of both decision-aware and general purpose approaches. This thesis presents the insights into how such algorithms can be built, based on theoretical analysis and experimental evidence.

1.1.2 Thesis structure

Before diving into the solutions to the problem outlined above, Chapter 2 presents a thorough overview of important definitions and algorithms in the field of reinforcement learning. This serves as the background for all further discussions in the thesis.

Chapter 3 presents insights into how instability emerges in value function learning. Experiments show that one of the major causes of unstable learning is diverging intermediate representations in the neural network architectures. Based on work conducted in Hussing et al. (2024), we can conclude that normalizing these representations has a vital stabilizing effect.

Chapter 5 dives deeper into the question of stabilizing representations for value function learning. While several different methods have been proposed in the literature, it remains an open question which model-based auxiliary tasks can be used as general-purpose methods to stabilize learning without deteriorating the quality of the representations for the decision making tasks. This chapter presents both theoretical and empirical evidence that latent self-prediction models provide benefitial features for this goal.

Chapter 4 investigates how the instability of value function learning can harm the training of decision-aware envrionment models. Empirical evidence is presented that suboptimal value function learning leads to a chicken-and-egg problem: without a good value function, we cannot train a good decision-aware model, but without a good decision-aware model, we cannot improve our value functionm To escape this cycle, we can modify a general purpose MSE loss with a sensitivity score derived from the current value function estimate.

Chapter 6 addresses additional issues with the value-aware model learning framework and show how it can lead to incorrect value and model estimates. One important source of problems is a subtle error term that appears when using common value-aware losses with stochastic models. This problem can be resolved by modifying the loss function to account for erroneous terms. Furthermore this chapter discusses how latent model architectures, such as those in presented in Chapter 5 can be used to learn strong value-aware models.

Chapter 7 presents an empirically strong algorithm that combines the ideas presented in this thesis. Using a latent value-aware model with well regularized features addresses a final important problem in value function learning: off-policy generalization. With a learned model, counterfactual questions of the form "what would have happened if the agent had taken action a' in a past state x" can be answered, which empirically has a strong stabilizing effect on value function learning.

Finally, this thesis concludes with **??** where we review the findings and discuss remaining questions and promising directions for future work.

Chapter 2

Background

Missing citations

To develop the core algorithms and proofs of the thesis, we need a brief survey of the fundamentals of Reinforcement Learning (RL), Model Based Reinforcement Learning (MBRL), and representation learning. These fields form the basis for the research presented in this thesis, and the methods and results are built on the established knowledge in these areas.

2.1 General mathematical notation and definitions

To define the concepts used in Machine Learning (ML) and especially in RL we require some mathematical language. Although this thesis does not aim for the level of rigor of a mathematical textbook, some care needs to be taken nonetheless to be precise with terminology and definitions.

When talking about sets, we will generally use italic capital letters such as $\mathcal{X}$ or $\mathcal{A}$, except for canonical terms such as the symbol $\mathbb{R}$ used for the real numbers. All sets used in this thesis are assumed to be measurable with standard measures, such as a counting measure in the case of a finite set, or a Borel measure in other cases.

Functions are denoted following standard terminology as lower case letters, and operators use upper case letters. We will write $M(\mathcal{X})$ to denote the set of possible distributions over the set $\mathcal{X}$ equipped with an appropriate σ-algebra. Functions that map to $M(\mathcal{X})$ are called probability kernels by convention and play an important role in the definition of RL. For a probability kernel $\mathcal{P} : \mathcal{X} \to \mathcal{Y}$ we will use the standard notational shorthand $\mathcal{P}(y|x)$ to denote the probability (density) of y under the distribution $\mathcal{P}$.

Given a suitable probability (density) $p(x)$ for the probability space $(\Omega, \mathcal{F}, p)$, we will de-

note the expectation of a random variable $X : \Omega \to \mathbb{R}^n$ drawn from p as $\mathbb{E}_{X\sim p}[X] = \int_\Omega X(\omega)\mathrm{d}p(\omega)$. This makes it clear from what distributions random variables are drawn when several distributions are involved such as in the case of the Kullback-Leibler Divergence (KL). In a discrete scenario, the Lesbeque integral can be simplified to $\sum_{x\in\mathcal{X}} p(x)x$, and in the case of regular real vector valued distributions such as the Gaussian we will similarly write $\int p(x)x\mathrm{d}x$ and forgo the full measure theoretic treatment. In these cases we will also not distinguish between random variables X and members of the underlying spaces $x \in \mathcal{X}$ and write x for simplicity.

2.2 The Markov Decision Process

RL is a mathematical problem formulation to model the concept of an agent gradually learning from interactions with its environment. At the heart of this formulation is the **Markov Decision Process (MDP)**. The notions presented here draw heavily from standard textbooks on the subject, as well as thesis.

Definition 1 (Markov Decision Process)**.** *An MDP is a tuple $(\mathcal{X}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where $\mathcal{X}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{P} : \mathcal{X} \times \mathcal{A} \to M(\mathcal{X})$ is a transition probability kernel, $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is a reward function, and $\gamma \in [0, 1]$ is a discount factor.*

These mathematical objects describe a process in which an agent interacts with an environment over a sequence of discrete time steps. An agent is said to be in a state $x \in \mathcal{X}$ at time t, where it takes an action $a \in \mathcal{A}$. The state at $t = 0$ is sampled from a starting state distribution p_0. Note that the state encompasses both the environment's state and the agent's internal state, such as memory or other information. The transition kernel is then used to determine a possible next state $x_{t+1} \in \mathcal{S}$ which the agent transitions to by drawing a sample from $\mathcal{P}(s, a)$. In the case of discrete state-action sets, these are simple discrete distributions, while some additional care is needed in the case of continuous state-action spaces to deal with the measure-theoretic properties of the state space. These are elaborated on in the appendix. At every timestep, after choosing an action a in state x, the agent receives the reward $r(s, a)$.

The agent's strategy for choosing actions is described by its policy, which describes the likelihood of the agent choosing different actions in state x.

Definition 2 (Policy)**.** *A policy $\pi : \mathcal{X} \to M(\mathcal{A})$ is a function that maps from states to distributions over actions.*

Equipped with all components of the MDP, we can now define trajectories and value functions.

Definition 3 (Trajectories)**.** *Let $M = (\mathcal{X}, \mathcal{A}, \mathcal{P}, r, \gamma)$ be an MDP. We call a set of states and actions $\tau^n := (x_0, a_0, \ldots, x_{n-1}, a_{n-1})$ a trajectory of length n. When n is omitted, we will generally assume a trajectory of infinite length. The distribution of a trajectory that is*

generated by an agent following π in M is defined as

$$p_{M,\pi}(\tau) = p_0(x_0)\pi(a_0|x_0)\prod_{i=1}^{n-1}\pi(a_i|x_i)\mathcal{P}(x_i|x_{i-1},a_{i-1}).$$

When the starting state is fixed, we will write

$$p_{M,\pi}(\tau|x_0) = \pi(a_0|x_0)\prod_{i=1}^{n-1}\pi(a_i|x_i)\mathcal{P}(x_i|x_{i-1},a_{i-1}).$$

In an analogous manner, we can also fix the starting state-action pair to obtain $p_{M,\pi}(\tau|x_0,a_0)$. We use $\mathcal{T}^n$ to denote the set of all possible trajectories of length n in M, and we will use τ_x to denote a trajectory starting from x. We denote the discounted reward associated with a trajectory as $r(\tau) = \sum_{i=0}^{n-1}\gamma^i r(x_i,a_i)$.

In cases where we are dealing with both timestep indices and sample indices, we will write $x_i^{(t)}$ where t is the timestep and i is the sample index. Often, the choice of MDP and policy is clear and we will omit the subscripts $\pi, \mathcal{P}$ from the notation to avoid clutter.

This definition allows us to easily define the value function of a policy.

Definition 4 (Value function)**.** *Let $M = (\mathcal{X}, \mathcal{A}, \mathcal{P}, r, \gamma)$ be an MDP. The policy value function $V^\pi : \mathcal{X} \to \mathbb{R}$ is equal to the expected discounted reward of infinite trajectories generated by an agent following policy π in M, which is equal to*

$$V^\pi(s) = \mathbb{E}_{\tau\sim p_{M,\pi}(\cdot|s)}\left[r(\tau)\right].$$

The policy state-action value function $Q^\pi : \mathcal{X}\times\mathcal{A} \to \mathbb{R}$ is defined as

$$Q^\pi(x,a) = \mathbb{E}_{\tau\sim p_{M,\pi}(\cdot|x,a)}\left[r(\tau)\right].$$

This definition has an important caveat – it assumes that the expectation exists and is finite. For example, if the sequence of rewards over a trajectory grows such that $r(x_n,a_n)/\gamma^n > 1$, the summation to compute $r(\tau)$ does not converge. To ensure this, we will generally assume that the reward is bounded so that for all state and action pairs $r_{\min} \le r(x,a) \le r_{\max}$. However, there are canonical problems such as the LQR problem for which this is not the case and some policies do not have a (finite) value function.

The goal of the agent in the standard MDP framework is to find an optimal policy π^* from a set of possible policies Π which maximizes the future discounted reward when starting from states sampled from p_0. This is equivalent to choosing

$$\pi^* \in \arg\max_{\pi\in\Pi} \mathbb{E}_{x\sim p_0}\left[V^\pi(x)\right].$$

The methods for finding such a policy are discussed in Section 2.3.

2.2.1 Occupancy distributions

It is often of interest to characterize the states visited under a policy without considering their temporal ordering. Sometimes we only care about how likely a state is to appear in any trajectory, not when it appears. We can formalize this notion by looking at the n-step transition probabilities. These are marginals of the trajectory probabilities.

Definition 5 (N-step Transitions)**.** *Let M be an MDP and π be a policy. The 1-step distribution $p^1(x, a|s_0)$ is defined as*

$$p^1_{M,\pi}(x, a|x_0, a_0) = \int_{\mathcal{A}} \pi(a|s)p(s|x_0, a_0)\mathrm{d}\pi(a_0|x_0).$$

The n-step transition is defined recursively as

$$p^n_{M,\pi}(x, a|x_0) = \int_{\mathcal{X},\mathcal{A}} \pi(s|a)p(x|x_{t-1}, a_{t-1})\mathrm{d}p^{n-1}(x_{t-1}, a_{t-1}|x_0).$$

Analogous to the trajectory case, $p^n(x, a|x_0, a_0)$ is the n-step probability resulting from the state-action pair (x_0, a_0).

Summing up and discounting n-step transitions results in the discounted state-action occupancy measure.

Definition 6 (Discounted state-occupancy measure)**.** *Let M be an MDP and π be a policy. The discounted state occupancy measure $\rho_{M,\pi}(x, a|x_0, a_0)$ is the distribution constructed as*

$$\rho_{M,\pi}(x, a) = \frac{1}{1-\gamma}\sum_{i=1}^{\infty} \gamma^i p^i(x, a|x_0, a_0).$$

Finally, there are MDPs where the sequence $p^n_{M,\pi}$ converges in distribution for $n \to \infty$. If this distribution is independent of the start state-action pair, we call it the stationary state-action distribution and write $\mu_{M,\pi}(x, a)$.

2.2.2 State and action representations

In the most general formulation of the MDP presented here, the states and actions are abstract collections of unstructured mathematical objects. In practice however, a state and an actions *representation* are crucial for the success of an RL algorithm. The representation of the state and action space can be seen as a mapping from the abstract state space to a concrete representation space, such as a vector of real numbers or an image.

A state observation function maps each state $x \in \mathcal{X}$ into a representation space $\mathcal{Y}$, often one that is equipped with structure such as $\mathbb{R}^n$. When we need to explicitly talk about such

a function, we will denote it as $o : \mathcal{X} \to \mathcal{Y}$. Common observation functions map discrete states to one-hot vectors, or follow more complex schemes such as tiling activations.

If the state observation function is not invertible, the resulting problem is called a Partially Observable Markov Decision Process (POMDP). In this framework, an agent cannot discern from the observation alone in which state it currently is, and needs to use other information, such as the past history of observations. In general, this thesis will stick to the realm of fully observable MDPs and steer clear of POMDPs. Nonetheless, they are an important class of problems in their own right, especially for modelling real world phenomena in which full observability is rarely if ever given.

It is not uncommon for an MDP problem to have more than one standard observation. For example, in many robotic control tasks, the state can either be represented as a set of joint angles and velocities, or by a camera image of the robot, or even a combination thereof! Often, changing the observation space can make a problem much harder or simpler to learn in practice.

One simple example of this is the representation of angles, a common problem in many real-world applications such as robotics. An angle θ can be represented as a numerical value $\theta \in [0, 2\pi)$. However, this representations has a discontinuity. For example, adding a small angle $\delta\theta$ to θ will generally result in a new angle $(\theta + \delta\theta) \bmod 2\pi$ with a discontinuity. If the same angle is instead represented as the vector $[\sin(\theta), \cos(\theta)]$, addition does not result in a discontinuous functions. However, this has the drawback that not all 2d vectors denote valid angles.

The problem of representing a state in a way that is beneficial to the goal of learning a good policy is generally called *representation learning*, and will be a major focus of this thesis.

2.3 Reinforcement Learning

> An optimal policy has the property that whatever the initial state and initial decisions are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decisions.
>
> Bellman 1953

Now that the problem of finding an optimal policy is established, we need algorithms to solve it. In this section, we review the basic concepts of RL that are used to obtain optimal policies. Given the scope of this thesis, we will focus solely on value driven methods that learn value functions and use these to improve upon the policy.

2.3.1 Reinforcement Learning in Tabular Domains

When the state and action sets are finite, all relevant components of the MDP problem can be represented by vectors and matrices. The policy-conditioned transition kernel $\mathcal{P}^\pi$ becomes a matrix of shape $|\mathcal{X}| \times |\mathcal{X}|$, and the reward and state value function a vector of shape $|\mathcal{X}|$.

Following standard Markov chain notation, any distribution over states is written as a row vector $x = [p(x_1], \ldots, p(x_n)]$ and the transition probability is

$$\left[\sum_{i=1}^{n} \mathcal{P}(x_0|x_i), \ldots, \sum_{i=1}^{n} \mathcal{P}(x_n|x_i)\right] = x\mathcal{P}^\pi.$$

The expected reward over a state distribution is also easily expressed as an inner product $\langle x, r\rangle$. We will frequently make use of the fact that $V^\pi = \sum_{i=0}^{\infty} \gamma \mathcal{P} r = (I - \gamma \mathcal{P}^\pi) r$.

2.3.2 Bellman optimality and Bellman operators

When looking at the definition of the policy value function, we can quickly see that it is recursive. We can decompose the value function as

$$V^\pi(x) = \mathbb{E}_{a\sim\pi(\cdot|x)} \left[r(x,a) + \gamma \mathbb{E}_{x'\sim\mathcal{P}(\cdot|x,a)} \left[V^\pi\left(x'\right)\right]\right].$$

This decompositions allows us to use one of the central concepts of reinforcement learning: the Bellman Optimality Principle, cited in the epigraph of this section. Assuming that an agent will follow the current policy in the subsequent state, an agent can improve its policy based on the current value function by computing

$$\pi_{\text{new}}(x) = \arg\max_{a\in\mathcal{A}} r(x,a) + \gamma \mathbb{E}_{x'\sim\mathcal{P}(\cdot|x,a)} \left[V^\pi\left(x'\right)\right].$$

The corresponding value function then becomes

$$V^{\pi_{\text{new}}}(x) = \max_a r(x,a) + \gamma \mathbb{E}_{x'\sim\mathcal{P}(\cdot|x,a)} \left[V^\pi\left(x'\right)\right] \geq V^\pi(x).$$

By greedily picking the best action under the one step reward and the *current* policies value function, the agent is guaranteed to receive at least the same future reward in each state as previously. This lays the foundation for a whole class of methods that all follow the same loop: pick a policy, compute its value function, improve the policy based on the value function, and repeat. The standard examples of such algorithms for tabular domains are Policy Iteration (PI) and Value Iteration (VI).

Policy Iteration In policy iteration, we begin by fixing a policy and computing its value function. DOing this takes advantage of the recursive nature of the function by the way of a fixed point iteration. Given any function F, we can define the Bellman operator as follows.

Definition 7 (On-policy Bellman Operator)**.** *Let M be an MDP, π be a policy, and $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ any function mapping state-action pairs to real numbers. The Bellman Operator $\mathcal{T}^\pi$ is defined as*

$$[\mathcal{T}^\pi Q](x,a) = r(x,a) + \gamma \mathbb{E}_{x' \sim \mathcal{P}(\cdot|x,a)} \left[\mathbb{E}_{a' \sim \pi(x')} \left[Q(x',a') \right] \right]$$

When starting with any initial function Q, repeatedly applying the Bellman Operator converges to The true policy state-action value function Q^π is the fixed point of this operator $[\mathcal{T}]$ sequence $[\mathcal{T}^\pi Q^\pi] = Q^\pi$.

With this Q function, we can define our new policy as $\pi^{\text{new}}(x) \leftarrow \arg\max_{a \in \mathcal{A}} Q(x,a)$ and then repeat the process. This can be proven to converge to the optimal policy π^*, by noticing that we have guaranteed improvement in π and that the process cannot get stuck in a local optimum. For a full proof, refer to Farahmand (2021).

Value Iteration Instead of waiting for our Bellman iteration to complete for every policy, we can instead greedily update the policy at every step. This leads to the so called Bellman optimality operator.

Definition 8 (Bellman Optimality Operator)**.** *Let M be an MDP, π be a policy, and $V : \mathcal{X} \to \mathbb{R}$ any function mapping state-action pairs to real numbers. The Bellman Optimality Operator $\mathcal{T}^*$ is defined as*

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

This again can be shown to converge directly to the Q function of the optimal policy.

2.3.3 Parametric value function learning

As the state-action space of an MDP grows larger, keeping an explicit record of each state's value in a long vector becomes less and less feasible. Memory requirements grow linearly with the size of the state-action space. Therefore we have to make use of approximations which can be stored efficiently in memory. In this thesis, we are foremost interested in parametric approximations.

Definition 9 (Parametric approximation)**.** *A parametric approximation of a value function is a combination of a k-dimensional* represention function *$\phi : \mathcal{X} \to \mathbb{R}^k$ and a weight vector $\omega \in \mathbb{R}^k$. Representation can be* fixed *or parameterized themselves with a vector of weights*

$\theta \in \mathbb{R}^l$. *When a learning algorithm updates the parameters of a parameterized representation, such a representation is called* learnable. *The value function is expressible as*

$$V(x|\theta, \omega) = \phi_\theta(x)^\top \omega.$$

However, once we have parameterized function, we cannot simply apply the Bellman operators in closed form as we did to obtain VI and PI in the tabular domain. To obtain the optimal weights of a parametric value function, we require a learning algorithm. While there is a huge variety of such algorithms, many currently popular ones revolve around the same core concept, approximate value iteration.

Loss functions for Value Learning

In machine learning, the most common way to find the optimal parameters of a function is by minimizing a loss function l over a dataset. Let $D = \{(x_1, y_1) \ldots, (x_n, y_n)\}$ be some dataset and $\mathcal{L}(f_\theta, D) = \sum_{i=1}^n l(y_i, f(x_i))$. Then the optimal function f^* from some function class $\mathcal{F}$ is

$$f^* = \arg\min_{f \in \mathcal{F}} \mathcal{L}(f_\theta, D).$$

To obtain it, we normally perform Gradient Descent (GD), although GD can only be proven to converge to local minima. The value function could in principle be obtained from computing targets from on-policy trajectories from the model, so that

$$\mathcal{L}(\hat{V}, \{x_i, \tau_{x_i}\}_n) = \left(r(\tau_{x_i}) - \hat{V}(x_i)\right)^2.$$

Such an approach is called a *Monte Carlo* approach, as it relies on a Monte Carlo or sample-based approximation of the value function.

The Double Sampling Issue in Bellman Residual Minimization However, this does not take advantage of the Bellman decomposition of the value function. To do this, we need to define a loss function that depends on the current value estimate in the target as well. Naively, we might want to try and minimize a loss function of the form

$$\mathcal{L}_{\text{res}}\left(\hat{V}, \{x_i, r_i, x_i'\}_n\right) = \left(\hat{V}(x_i) - r - \gamma \hat{V}(x_i')\right)^2.$$

This is called *Bellman Residual Minimization*, and we can show that it does not lead to a correct value function estimate. This result is well known in the reinforcement learning literature. Let us broefly review the proof here as similar techniques will be relevant to show issues with more advanced loss functions in later chapters of this thesis.

Proposition 1 (The Double Sampling Problem)**.** *Let D be a dataset of $\{x_i, r_i, x_i'\}$ tuples*

sampled from the stationary distribution of some MDP M with transition kernel $\mathcal{P}$ and fixed policy π. There exist function classes $\mathcal{V}$ that include V^π, the ground truth value function for $\mathcal{P}^\pi$, but for which

$$V^\pi \notin \arg\min_{\hat{V}\in\mathcal{V}} \mathbb{E}_D\left[\mathcal{L}_{\text{res}}(\hat{V}, \{x_i, r_i, x_i'\})\right].$$

The proof is relatively straight forward. We need to show that the intended target, V^π, does not properly minimize the loss. To do this, we decompose the loss function as follows

$$\begin{aligned}
\mathbb{E}_D\left[\mathcal{L}_{\text{res}}(\hat{V}, \{x_i, r_i, x_i'\})\right] &= \mathbb{E}_D\left[\frac{1}{n}\sum_{i=1}^{n}\left(\hat{V}(x_i) - r_i - \gamma\hat{V}(x_i')\right)^2\right] \\
&= \mathbb{E}_D\left[\left(\hat{V}(x_i) - r_i - \gamma\hat{V}(x_i')\right)^2\right] \\
&= \mathbb{E}_D\left[\left(\hat{V}(x_i) - V^\pi(x_i) + V^\pi(x_i) - r_i - \gamma\hat{V}(x_i')\right)^2\right] \\
&= \mathbb{E}_D\left[\left(\hat{V}(x_i) - V^\pi(x_i)\right)^2\right] + \mathbb{E}_D\left[\left(V^\pi(x_i) - r_i - \gamma\hat{V}(x_i')\right)^2\right] \\
&\quad - \mathbb{E}_D\left[\left(\hat{V}(x_i) - V^\pi(x_i)\right)\left(V^\pi(x_i) - r_i - \gamma\hat{V}(x_i')\right)\right]
\end{aligned}$$

If we substitute the true value function V^π for $\hat{V}$, it is easy to see that the first and last terms disappear. However, we are left with a variance like term

$$\begin{aligned}
\mathbb{E}_D\left[\mathcal{L}_{\text{res}}(V^\pi, \{x_i, r_i, x_i'\})\right] &= \mathbb{E}_D\left[\frac{1}{n}\sum_{i=1}^{n}\left(V^\pi(x_i) - r_i - \gamma V^\pi(x_i')\right)^2\right] \\
&= \mathbb{E}_D\left[\left(\underbrace{V^\pi(x_i) - V^\pi(x_i)}_{=0}\right)^2\right] + \mathbb{E}_D\left[\left(V^\pi(x_i) - r_i - \gamma V^\pi(x_i')\right)^2\right] \\
&\quad - \mathbb{E}_D\left[\underbrace{\left(\hat{V}(x_i) - V^\pi(x_i)\right)}_{=0}\left(V^\pi(x_i) - r_i - \gamma\hat{V}(x_i')\right)\right] \\
&= \mathbb{E}_D\left[\left(V^\pi(x_i) - r_i - \gamma V^\pi(x_i')\right)^2\right] \neq 0.
\end{aligned}$$

Our formal statement requires a function that minimizes this loss better than the ground truth value function in our function class. The existence of such a function can easily be shown by example.

If we had two independent next state samples x_i', we could estimate the magnitude of the variance remainder term and correct the loss. But in the standard RL setting, we assume

strict sequential interaction. So in general we can not rely on having two independent samples for each state whose value we want to estimate.

Target Network Updates Instead of naively minimizing the predicted difference between our current value function estimate, we only need a minimal change to make the loss function work. We cannot update the parameters of both the value estimate $\hat{V}$ and the bootstrapped Bellman estimate $R + \gamma\hat{V}$. So instead, we make use of an independent copy of $\hat{V}_{\text{targets}}$, often called a target network, and use

$$\mathcal{L}_{\text{res}}\left(\hat{V}, \hat{V}_{\text{target}}, \{x_i, r_i, x_i'\}_n\right) = \frac{1}{n}\sum_{i=1}^{n}\left(\hat{V}(x_i) - \left[r + \gamma\hat{V}_{\text{target}}(x_i')\right]_{\text{sg}}\right)^2$$

as our loss. The notation $[\cdot]_{\text{sg}}$ refers to a *stop gradient* operation, which clarifies that the loss function is not used to update the later part of the equation.

Different algorithms chose different approaches to obtain the target estimate $\hat{V}_{\text{target}}$. The simplest two variants are to either replace $\hat{V}_{\text{target}}$ at *every* update step, or instead to wait until $\hat{V}$ has converged to an optimum and update then. However, the former version can prove to be very unstable in practice, will the later can be prohibitively slow. Instead, most common algorithms either use a fixed period, e.g. 1000 steps, between target value updates, or use a Polyak average of the parameters $\theta_{\text{target}} \leftarrow (1-\alpha)\theta_{\text{target}} + \alpha\theta$ with a small value for α.

On- and off-policy learning After establishing a loss function that can be used to learn parametric value functino approximations, we also need data to train on. Here, two major strategies dominate: on-policy and off-policy learning.

In *on-policy* learning, all data comes from a single policy, the one for which we are attempting to estimate the value. While this is generally the much more stable case, it also requires dramatically more samples than the alternative, as all samples need to be discarded once we change the policy. In principle, old samples can be used if we apply a sampling correction term such as importance sampling. But even in this case, the importance weights quickly become too small for old samples to have an impact on the loss. As this thesis is interested in *sample efficient* learning, we instead focus on off-policy learning.

Off-policy learning allows us to use samples from old policies, but introduces several algorithmic challenges. Firstly, we are forced to rely on state-action value functions (Q) instead of state value functions (V), as we cannot improve our policy otherwise. More importantly, many algorithms can diverge when used with off-policy samples, even when their on-policy counterparts converge neatly. The (in-)stability of *off-policy* learning is the focus of Chapter 7.

2.4 Deep Reinforcement Learning in Continuous Action Spaces

Current Deep Reinforcement Learning (DRL) methods use several additional mechanisms to ensure stable convergence of the RL agent's value function and policy. Here, we take a brief look at the Twin Delayed Deep Deterministic Policy Gradient (TD3) and Soft Actor Critic (SAC) algorithms, as they are used throughout the empirical work presented in this thesis.

Both TD3 and SAC are actor-critic algorithms, which interleave value function and policy updates. They are built for continuous state-action spaces and estimate state-action value functions. In addition, they use parameterized policies, as the policy update $\pi(\cdot|s) \leftarrow \arg\max_{a \in actions} Q(s,a)$ cannot be solved in closed form. Actor-critic algorithms therefore replace the closed form maximization with another gradient based update. In the case of TD3 and SAC, this is the Deterministic Policy Gradient (DPG) (Silver, Lever, et al. 2014)

$$\theta_\pi \leftarrow \theta_\pi + \alpha \nabla_{\theta_\pi} Q(s, \pi(s, \theta_\pi)).$$

While TD3 uses a deterministic policy $\pi : \mathcal{X} \to \mathcal{A}$, SAC uses a probabilistic Gaussian policy and computes the gradient using the re-parameterization trick

$$\nabla_\theta \mathbb{E}_{y \sim \mathcal{N}(\mu_\theta(x), \sigma_\theta(x))}\left[f(y)\right] \approx \sum_{i=1}^{N} \frac{\mathrm{d}\, f(\mu_\theta(x) + \epsilon_i \sigma_\theta(x))}{\mathrm{d}\, \mu_\theta(x) + \epsilon_i \sigma_\theta(x)} \frac{\mathrm{d}\, \mu_\theta(x) + \epsilon_i \sigma_\theta(x)}{\mathrm{d}\, \theta}, \quad \epsilon_i \sim \mathcal{N}(0,1)$$

The main innovation of TD3 is the introduction of the twinned critic to counterbalance overestimation in the critic estimation. The overestimation problem stems from the fact that the actor is updated to maximize the critic's prediction. If the critic estimate is noisy, the selected action's value will in general be larger than the true value, as

$$\mathbb{E}_{\epsilon \sim q}\left[\max_a Q(s,a) + \epsilon\right] \geq \max_a Q(s,a).$$

Intuitively, the max optimization with regard to the critic will optimize towards erroneously overestimated values.

To counterbalance this, Fujimoto, Hoof, and Meger (2018) introduce two independent critic estimates Q_1 and Q_2. The double critic's loss function is

$$\mathcal{L}(Q_i|s,a,r,s',\pi) = \left(Q_i(s,a) - \left[r + \gamma \min_{j \in [1,2]} Q_j(s', \pi(s'))\right]\right)^2.$$

The actor update also maximizes this minimum over both critics

$$\mathcal{L}(\pi|s, Q_1, Q_2) = -\min_{j\in[1,2]} Q_i(s, \pi(s)).$$

With some assumptions on the noise this max-min structure can be proven to prevent overestimation. However, as will be discussed in Chapter 3, in practice this trick is helpful but insufficient by itself to prevent overestimation. In addition to statistical error, the optimization process of the critic can be unstable and lead to overestimation.

SAC uses the same double critic structure of TD3 and modifies the critic to optimize a maximum entropy objective. For this, the critic target is modified to include the stochastic policies entropy over the next state

$$\mathcal{L}(Q_i|s, a, r, s', \pi) = \left(Q_i(s, a) - \left[r + \gamma \min_{j\in[1,2]} Q_j(s', a') - \log \pi(a'|s')\right]\right)^2, \quad a' \sim \pi(\cdot|s').$$

The policy objective function is similarly modified

$$\mathcal{L}(\pi|s, Q_1, Q_2) = -\min_{j\in[1,2]} Q_i(s, a) + \log \pi(a|s), \quad a \sim \pi(\cdot|s).$$

This maximum entropy modification can be interpreted in the framework of "control-as-inference" and has been shown to be empirically more stable than TD3. However, it is also computationally more expensive, and so several algorithms proposed in this thesis will use TD3 instead of SAC.

2.5 Model-Based Reinforcement Learning

Many reinforcement learning algorithms compute their value functions and policies directly based on past data from the environment. In MBRL on the other hand, a predictive *model*[1] of the environment is trained and used in the RL algorithm. This model can be used to augment the learning, either by providing additional data as in the Dyna architecture (Richard S Sutton 1990), or by providing gradient estimates (Hafner et al. 2020; Amos et al. 2021), or simply to enhance the representation learning as discussed in Chapter 5.

The most common kind of model is a *forward prediction model*. This is a parameterized function, such as a neural network, that takes a state-action pair as an input and predicts the next state. Simple extensions to these include multi-step models which use sequences of actions to predict short-horizon trajectories and stochastic models which parameterize

[1] *Model* is a notoriously ambiguous term in machine learning. In the context of this thesis, the word *model* is used solely to describe an approximation of a transition function, reward function, or a similar object, while neural networks and other architectures will be referred to as *function approximators* or simply *functions*.

distributions over next states.

Definition 10 (Types of environment models)**.** *A probabilistic* environment model *is a parameterized distribution $\hat{p}_\theta(x'|x,a)$ that approximates an MDP's transition kernel. All relevant definitions from Section 2.2 can be extended for environment models.*

A deterministic *environment model is a function $f : \mathcal{X} \times \mathcal{A} \to \mathcal{X}$ that directly maps to a next state.*

A latent *environment model is a combination of an embedding function $\phi : \mathcal{X} \to \mathcal{Z}$ that maps from the state space $\mathcal{X}$ to a latent space $\mathcal{Z}$, and a latent dynamics model $f_\mathcal{Z} : \mathcal{Z} \times \mathcal{A} \to \mathcal{Z}$. Unless specified otherwise, we assume that $\mathcal{Z} = \mathbb{R}^k$, where k is the dimensionality of the latent space. Trajectories from a latent model are sampled conditioned on the embedding of an initial state x_0.*

There are several other types of models, such as inverse dynamics models which seek to predict actions given a state and its successor, but in this thesis, we will focus on the more common forward predictive models.

2.5.1 Standard training algorithms and parameterization

Most commonly, models are trained by obtaining a dataset $\mathcal{D}$ of transition tuples consisting of states x, actions a, reward r, and next states x'. Often these datasets are obtained online during interaction with the environment, but they can also come from i.e. demonstrations or other offline sources.

Given a dataset, an approximate model $\hat{p}$ can be trained using a Maximum Likelihood Estimation (MLE) objective

$$\max_{\hat{p}} \frac{1}{N} \sum_{i=1}^{N} \log \hat{p}(x'_i|x_i, a_i).$$

For example, a Gaussian environment model, such as the one used by Chua et al. (2018), Janner et al. (2019a), and Voelcker, Liao, et al. (2022), uses the Gaussian negative log likelihood over a dataset as the loss function

$$\mathcal{L}_{\text{model}}(\mu_\theta, \Sigma_\theta; D) = -\frac{1}{N} \sum_{i=1}^{N} \left[\log |\Sigma_\theta(x_i, a_i)| + (x'_i - \mu_\theta(x_i, a_i))^\top \left(\Sigma_\theta^{-1}(x_i, a_i) \right) (x'_i - \mu_\theta(x_i, a_i)) \right].$$

For deterministic models, a common objective is the Mean Squared Error (MSE)

$$\mathcal{L}_{\text{model}}(f_\theta; D) = \frac{1}{N} \sum_{i=1}^{N} (x'_i - f_\theta(x_i, a_i))^2.$$

These methods are commonly referred to as *reconstruction objectives* or *observation prediction* objectives, as the learned model predicts x' as it is provided by the environment or dataset. Later in this thesis, we will discuss *latent space models* more thoroughly as an alternative.

In addition to the next state, environment models are often trained to predict the reward as well, as the reward function is not generally assumed to be known. Unless noted otherwise, we will generally assume that the reward function is predicted using the model whenever the model is used.

2.5.2 DYNA

The DYNA framework Richard S Sutton 1990 is one of the most common ways of using an environment model in RL. The main idea is very simple: instead of only using the real environment for generating transition samples $\{x, a, r, x'\}$ we use the model as well. In this setup, the use of the model is invisible to the RL algorithm which computes the value function and policy. This makes it very flexible, as we can freely vary the model training and RL algorithm without changing the overall framework. **??** discussed how using a model in a DYNA framework impacts the empirical performance of modern DRL algorithms.

Add nice picture and a bit more detail here

2.5.3 Objective mismatch phenomenon

One of the most important insights that form the basis of this thesis is the fact that model training objectives such as MLE are not aligned with the objective of *training a good model for RL*. As a simple example, assume a Dyna setting where a sample is collected from a deterministic model and has an error ϵ. A value function based method will use the model sample to compute a biased bootstrap target

$$V_k(s, a) = r(s, a) + \gamma V_{k-1}(f(s, a)+\epsilon).$$

The impact of the modelling error on the value function therefore depends on the size of the error and the local behavior of the value function, not only on the absolute value of ϵ. We could colloquially say that not all errors are created equal.

As another illustrative example take a value function that only depends on a subset of all state observation dimensions. In this case, a large error in an irrelevant dimension has no consequence on the obtained policy, yet a maximum likelihood loss for the model cannot properly capture this behavior without prior handcrafted features.

Intuitively, we would like to have an objective that bounds the difference in the value

function estimate. We can motivate the use of MLE (such as the mean squared error for a Gaussian model with fixed variance) as a loss function by an upper bound

$$\sup_{V \in \mathcal{F}} |\langle p - \hat{p}, V \rangle| \leq ||p - \hat{p}||_1 \sup_{V \in \mathcal{F}} ||V||_\infty \leq \sqrt{\mathrm{KL}(p||\hat{p})} \sup_{V \in \mathcal{F}} ||V||_\infty$$

(taken from Farahmand, André Barreto, and Nikovski (2017)), where the inequality is Pinsker's inequality. However, this bound is loose and does not account for the geometry of the problem's value function or any other knowledge that the agent has collected. In our example above a mean squared error would penalize deviations equally by their L_2 norm without accounting for the relevance of the dimensions. This problem was termed the *objective mismatch* by (Lambert et al. 2020).

To refer to our goal of learning models which mitigate the mismatch, we will use the terms "decision-aware" and "value-aware" mostly interchangeably. Both were introduced by Farahmand, André Barreto, and Nikovski (2017), the former refers more generally to models which account for the downstream decision task, while the latter refers more concretely to models which account for errors in the value function target estimate.

Value-aware model learning

To address the model mismatch, (Farahmand, André Barreto, and Nikovski 2017) proposed *Value-aware Model Learning* (VAML), a loss function that captures the impact the model errors have on the one-step value estimation accuracy. The core idea behind VAML is to penalize a model prediction by the resulting difference in a value function. Given a distribution over the state-actions space μ and a value function V, it is possible to define a value-aware loss function $\mathcal{L}_V(\hat{p}, p, \mu)$:

$$\mathcal{L}_V(\hat{p}, p, \mu) = \int \mu(s,a) \left| \overbrace{\int p(s'|s,a)V(s')\mathrm{d}s'}^{\text{environment value estimate}} - \overbrace{\int \hat{p}(s'|s,a)V(s')\mathrm{d}s'}^{\text{model value estimate}} \right|^2 \mathrm{d}(s,a)$$

and its empirical approximation $\hat{\mathcal{L}}_V$ based on a dataset $D = (s_i, a_i, s'_i)_{i=1}^N$ of samples from μ and p

$$\hat{\mathcal{L}}_V(\hat{p}, \mathcal{D}) = \sum_{(s_i, a_i, s'_i) \in \mathcal{D}} \left| V(s'_i) - \int \hat{p}(s'|s_i, a_i)V(s')\mathrm{d}s' \right|^2 .$$

The main problem of this approach is that it relies on the value function, which is not known a priori while learning the model. This leads to the "chicken-or-egg" problem of decision aware learning and a central topic in this thesis.

If a good model depends on knowing the decisions and good decisions need to be learned

using the model, how can we learn a good model before knowing what the correct decision is?

In the original formulation by Farahmand, André Barreto, and Nikovski (2017), the value function is replaced with the supremum over a function space to enable analysis. While this fixes the chicken-egg problem of decision-aware model learning and works well in the case of linear value function spaces, finding a supremum for a function space parameterized by complex function approximators like neural networks is difficult. Furthermore, the supremum formulation is conservative and does not account for the fact that knowledge about the value function is gained over the course of exploration and optimization in a MBRL approach.

Instead of the supremum over a value function class, Farahmand (2018) introduced an extension of VAML where the supremum is replaced with the current estimate of the value function, *Iterative Value-Aware Model Learning* (IterVAML). In each iteration, the value function is updated based on the model, and the model is trained using the loss function based on the last iteration's value function. The author presents error bounds for both steps of the iteration, but did not test the algorithm to ascertain whether the presented error bounds are sufficient to guarantee a strong algorithm in practice. Furthermore, these work assume that both the Approximate Value Iteration and model learning procedure are conducted until they reach a small error at every step, which is often prohibitively expensive, or even impossible in case of neural network value function approximations. We will look at IterVAML more extensively in Chapter 4 and Chapter 6.

Abstract latent value models

Silver, H. v. Hasselt, et al. (2017) introduced the idea of a purely abstract latent space model in which the transitions is aligned with the reward and value functions of the environment. Their work considers an uncontrolled setting, in which an action-independent Markov Reward Process (MRP) is modelled. The goal of their system is to learn the reward function of this process, which can be seen as doing policy evaluation in an MDP with a fixed policy. The core difference of their approach to other model-based reinforcement learning approaches is that they learn an abstract transition model where the states do not have a one-to-one correspondence to the environment observations.

Extending their work, Oh, Singh, and H. Lee (2017) show how to build an abstract model of a fully controllable MDP and highlight how such a model can be used for planning in latent space with a tree search approach such as MCTS (Schrittwieser et al. 2020) or beam search. As we will see in Chapter 6, the formulations used by Silver, H. v. Hasselt, et al. (2017) and Schrittwieser et al. (2020) and IterVAML share close similarities, but the proposed value function learning objective provides a biased learning target for stochastic

environment models.

2.5.4 Tradeoffs between value-aware and general purpose methods

One major problem that motivates the work conducted in this thesis is that a single value function alone does not necessarily provide sufficient information about the transitions to learn precise models. As mentioned, reinforcement learning is fundamentally non-stationary in nature, which means that the current prediction task is not necessarily indicative of future prediction tasks.

This is apparent in a sparse reward setting: when the task requires exploration, the value function continues to predict 0 for all visited states until the agent has actually reached the goal. A purely value function driven model or representation can then collapse the whole state space into a single abstract state and potentially completely prevent further exploration. This phenomenon is partially discussed by Tomar et al. (2023), who show that many techniques for value-based abstractions fail in hard-exploration scenarios. Similarly, Kemertas and Aumentado-Armstrong (2021) shows that auxiliary training objectives strongly improve the performance of bisimulation-based representation learning, especially with sparse rewards. A related discussion in the the context of IterVAML will be presented in Chapter 4.

This suggests that there is a trade-off between general purpose model learning, such as successor features or sequential RSSMs, and task-specific methods such as IterVAML, bisimulation learning or MuZero. While the former ones are often more robust to shifts in the task and missing value information, they fail to be as efficient as the latter especially under resource constraints such as network size.

2.6 Representation learning in RL

The main goal of learning useful representations is to capture some desirable characteristic with regards to a prediction task in the latent space. This is often some form of smoothness or (Lipschitz)-continuity, or come optimal compression compression. Abel (2020) and Le Lan, Marc G Bellemare, and Castro (2021) provide excellent overviews of the general goals of representations in the context of RL.

One of the major components of the success of Deep Learning has been the surprising capability of neural networks to *learn* good representations for tasks such as classification (Bengio, Courville, and Vincent 2013). One of the most prominent examples are the structured latent spaces obtained by the word2vec algorithm, where training to predict (embeddings of) words leads to structured spaces that allow for meaningful vector arithmetic (Mikolov et al. 2013; Goldberg and Levy 2014).

The core challenges for representation learning in RL is that the tasks are fundamentally non-stationary in nature (Kumar et al. 2021; Nikishin, Schwarzer, et al. 2022). As the agent explores the environment, value function and policy continually change, meaning that both the input distribution and the prediction targets change over time. This has led to the establishment of research into *learning dynamics* (Lyle, Rowland, Dabney, et al. 2022; Lyle, Rowland, and Dabney 2022), *metric learning* (Ferns, Panangaden, and Precup 2011; André Barreto et al. 2017; Borsa et al. 2019; Le Lan, Marc G Bellemare, and Castro 2021), and *auxiliary tasks* (Jaderberg et al. 2017; Marc G Bellemare et al. 2019; Lyle, Rowland, Ostrovski, et al. 2021; Farebrother, Greaves, et al. 2023) to address these challenges. In this thesis, we will see how the learning dynamics of latent representations can be shaped by and used for model-based reinforcement learning.

Desirable properties of representations

Before discussing how representation learning in RL can be improved, it is necessary to characterize representations that leads to good performance in RL. In many classic RL approaches, that state space is taken to be a finite set of unordered states without any metric or distance between these except for the one induced by the transition function. Over such abstract states, one of the most commonly considered representations is a state aggregation. State aggregations are thoroughly discussed in Abel (2020), where three core desiderata are established: good state aggregations should be easy to compute, enable efficient learning, and allow the selection of high-value policies.

Beyond state abstraction, the topology and metric of a representation space can also be considered. Le Lan, Marc G Bellemare, and Castro (2021) and Le Lan, Tu, Oberman, et al. (2022) discuss the capabilities of representations to enable generalization of learned value functions over the state space by analyzing the induced topology and metric space from different representation learning targets. Their work allows us to consider more than the size of the induced representation space and consider the impact of the representation on the learning dynamics and generalization capabilities of different algorithms

Ghosh and Marc G Bellemare (2020) extends the analysis of representation beyond fixed properties such as metric spaces to consider the impact of the given representation on the stability of the learning task. Using tools from linear dynamical system, they show that some proposed representations provide more stable learning dynamics for TD-learning as others. We will take an in-depth look at analyzing linear systems in Chapter 5.

In this work, we draw on these characterizations to analyze both the usefulness of different representation learning approaches for representing good value functions, as well as analyzing whether they enable stable learning dynamics (and indeed, can themselves be learned in a stable manner).

Sadly, these desirable properties do not seem to emerge quite as easily in the course of RL training as in supervised learning, and a lot of recent work has focused on analyzing and stabilizing representation learning in RL.

General purpose representation learning and zero-shot RL

To obtain optimal representations for reinforcement learning tasks in a given environment, several different approaches have been proposed. Several of the leading approaches are inspired from linear algebra and graph theory and seek to model fundamental aspects of the transition matrix.

Broadly speaking, the transition matrix $\mathcal{P}$ and the resolvent matrix $(I - \gamma\mathcal{P})^{-1}$ can be approximated by different decompositions such as eigenvalue decompositions, singular value decomposition, or Schur decomposition (Ghosh and Marc G Bellemare 2020). These matrix approximations can then be used to obtain i.e. the largest eigenvectors which capture the long-term transition behavior of the underlying Markov chains. For example, the equality $V = (I - \gamma\mathcal{P})^{-1}r$ shows that if the resolvent matrix is well approximated, a representation can be obtained that can be used to approximate the value function of any reward.

The goal of obtaining well-performing representations has also been addressed with empirically motivated approaches. Many of these seek to construct so called *auxiliary tasks* (Jaderberg et al. 2017), which are additional prediction objectives or loss functions which are added to the learning objectives of the RL agent. Several methods simply seek to add next state prediction , either with an auto-encoder architecture (Jaderberg et al. 2017), contrastive (Laskin, Srinivas, and Abbeel 2020), or self supervised objectives (Gelada et al. 2019; Schwarzer, Anand, et al. 2021; Schwarzer, Rajkumar, et al. 2021; Tang, Z. D. Guo, et al. 2022). These methods tend to perform well especially in cases with high-dimensional observations, i.e. as pixel based environments such as Atari games. The main distinction between these methods and model-based RL is that the next state prediction is only used to improve the performance of an encoder function, the next state predictions are not directly used to improve the RL agents policy or value function estimation.

Beyond state prediction tasks, several works have also shown the advantage of using value function prediction of auxiliary or randomly generated rewards as representation learning targets (Lyle, Rowland, Ostrovski, et al. 2021; Farebrother, Greaves, et al. 2023). These commonly use the same approaches as off-policy value function learning, but instead of using the estimated value functions to derive a policy, the results are again discarded and only used for stabilization.

Chapter 3

Instability of Value Function Learning

> This chapter is based on Marcel Hussing, Claas A Voelcker, Igor Gilitschenski, Amir-massoud Farahmand, and Eric Eaton (2024). "Dissecting Deep RL with High Update Ratios: Combatting Value Divergence". In: *Reinforcement Learning Conference*.

To improve sample efficiency, contemporary work in off-policy deep reinforcement learning (RL) has begun increasing the number of gradient updates per collected environment step (Janner et al. 2019b; Fedus et al. 2020; Xinyue Chen, C. Wang, Zhou, and Keith W. Ross 2021; Hiraoka et al. 2022; Nikishin, Schwarzer, et al. 2022; D'Oro, Schwarzer, et al. 2023; Schwarzer, Obando Ceron, et al. 2023; Kim et al. 2023). As this update-to-data (UTD) ratio increases, various novel challenges arise. Notably, a recent study proposed the emergence of a *primacy bias* in deep actor critic algorithms, defined as "a tendency to overfit initial experiences that damages the rest of the learning process" (Nikishin, Schwarzer, et al. 2022). This is a fairly broad explanation of the phenomenon, leaving room for investigation into how fitting early experiences causes suboptimal learning behavior.

First approaches to tackle the learning failure challenges have been suggested, such as completely resetting networks periodically during the training process and then retraining them using the contents of the replay buffer (Nikishin, Schwarzer, et al. 2022; D'Oro, Schwarzer, et al. 2023). Resetting network parameters is a useful technique in that, in some sense, it can circumvent any previous optimization failures without prior specification. Yet it seems likely that a more nuanced treatment of the various optimization challenges in deep RL might lead to more efficient training down the line. Especially if the objective is efficiency, throwing away all learned parameters and starting from scratch periodically is counter-productive, for instance in scenarios where, keeping all previous experience is

infeasible. As such, we set out to study the components of early training that impair learning more closely and examine whether high-UTD learning without resetting is possible.

To motivate our study, we repeat the priming experiment of Nikishin, Schwarzer, et al. (2022), in which a network is updated for a large number of gradient steps on limited data. We show that during priming stages of training, value estimates diverge so far—and become so extreme—that it takes very long to unlearn them using new, counter-factual data. However, contrary to prior work, we find that it is not *impossible* to learn even after priming, it merely takes a long time and many samples. This sparks hope for our endeavor of smooth learning in high-UTD regimes. We show that compensating for the value function divergence allows learning to proceed. This suggests that the failure to learn does not stem from overfitting early data, which would result in correct value function on seen data, but rather from improperly fitting Q-values. We demonstrate that this divergence is most likely caused by prediction of out-of-distribution (OOD) actions that trigger large gradient updates, compounded by the momentum terms in the Adam optimizer (Kingma and J. Ba 2015).

The identified behavior, although triggered by OOD action prediction, seems to be more than the well-known overestimation due to statistical bias (Thrun and Schwartz 1993). Instead, we hypothesize that the problem is an optimization failure and focus on limiting the exploding gradients from the optimizer via architectural changes. The main evidence for this hypothesis is that standard RL approaches to mitigating bias, such as minimization over two independent critic estimates (Fujimoto, Hoof, and Meger 2018), are insufficient. In addition, using pessimistic updates (Fujimoto, Meger, and Precup 2019; Fujimoto and Gu 2021) or regularization (Krogh and Hertz 1991; Srivastava et al. 2014) to treat the value divergence can potentially lead to suboptimal learning behavior, which is why architectural improvements are preferable in many cases.

We use a simple feature normalization method (B. Zhang and Sennrich 2019; C. Wang et al. 2020; Bjorck, Gomes, and Weinberger 2022) that projects features onto the unit sphere. This decouples learning the scale of the values from the first layers of the network and moves it to the last linear layer. Empirically, this approach fully mitigates diverging Q-values in the priming experiment. Even after a large amount of priming steps, the agent immediately starts to learn. In a set of experiments on the dm_control MuJoCo benchmarks (Tunyasuvunakool et al. 2020a), we show that accounting for value divergence can achieve significant across-task performance improvements when using high update ratios. Moreover, we achieve non-trivial performance on the challenging dog tasks that are often only tackled using model-based approaches. We demonstrate comparable performance with the recently developed TD-MPC2 (N. Hansen, Su, and X. Wang 2024), without using models or advanced policy search methods. Lastly, we isolate more independent failure modes, giving pointers towards their origins. In Appendix **??** we list open problems whose solutions

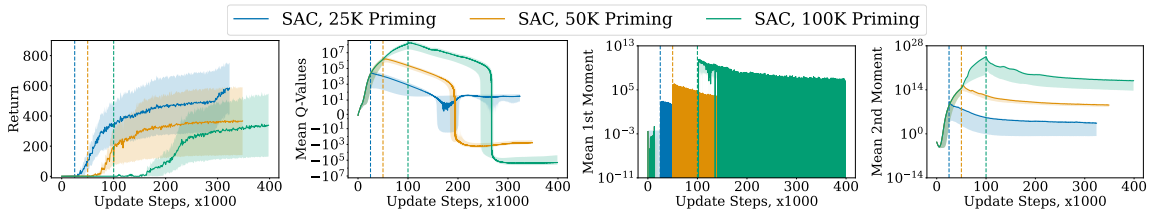

Figure 3.1: Return, in-distribution Q-values and Adam optimizer moments during priming for different lengths. Dotted lines correspond to end of priming. More priming leads to lower return and larger Q-value and optimizer divergence.

might illuminate other RL optimization issues.

3.1 Preliminaries

3.2 Investigating the effects of large update-to-data ratios during priming

As mentioned, the definition of the primacy bias is broad. To obtain a more nuanced understanding, we set out to re-investigate the early stages of high-UTD training. To do so, we repeat the priming experiment conducted by Nikishin, Schwarzer, et al. (2022).[1] We first collect a small amount of random samples. Then, using the SAC algorithm, we perform a priming step, training the agent for a large number of updates without additional data. After priming, training continues as usual. Prior results reported by Nikishin, Schwarzer, et al. (2022) suggest that once the priming step has happened, agents lose their ability to learn completely. We use the simple Finger-spin task (Tunyasuvunakool et al. 2020a) to study the root causes for this systematic failure and to examine if there are ways to recover without resets. In this section, we report means over five random seeds with standard error in shaded regions. Hyperparameters are kept consistent with previous work for ease of comparison.

3.2.1 An old acquaintance: Q-value overestimation

We first ask whether there is a barrier as to how many steps an agent can be primed for before it becomes unable to learn. We test this by collecting 1,000 samples and varying the number of updates during priming from 25,000 to 50,000 and 100,000. The results are presented in Figure 3.1.

We make two key observations. First, lower amounts of priming are correlated with higher

[1] For consistency with later sections, we use the ReLU activation here which can lead to unstable learning of other components. We repeat all the experiments with ELUs in Appendix **??** to provide even stronger support of our findings.

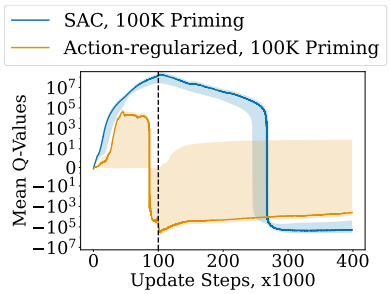

Figure 3.2: Priming with SAC and action regularization during priming. The latter lowers divergence.

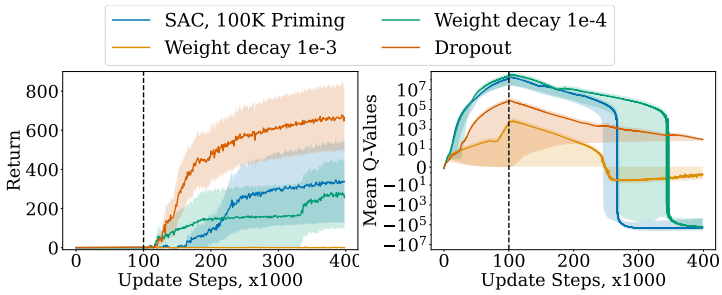

Figure 3.3: Return and Q-values of priming runs with weight decay and dropout. Results indicate that both regularizations mitigate priming to some extent but not sufficiently.

early performance. More precisely, it seems that many runs simply take longer before they start learning as the number of priming steps increases. Second, during priming the scale of the average Q-value estimates on observed state-action pairs increases drastically. We find that the Q-values start out at a reasonable level, but as priming goes on they eventually start to diverge drastically. Once the agent estimates very large Q-values, the final performance in terms of average returns deteriorates. We also observe that the second moment of the Adam optimizer (Kingma and J. Ba 2015) is correlated with the divergence effect. Optimizer divergence has been observed before as a cause of plasticity loss under non-stationarity (Lyle, Zheng, Nikishin, et al. 2023), but in our experiments the data is stationary during priming. We conjecture that the momentum terms lead to much quicker propagation of poor Q-values and ultimately to prediction of incorrect Q-values, even on in-distribution data.

After priming, there exist two cases: 1) either the Q-values need to be unlearned before the agent can make progress or 2) there is a large drop from very high to very low Q-values that is strongly correlated with loss in effective dimension of the embedding, as defined by Yang et al. 2020 (see Appendix **??**). In the second case, rank can sometimes be recovered upon seeing new, counter-factual data and the network continues to learn. Yet, sometimes the agent gets stuck at low effective dimension; a possible explanation for the failure to learn observed in the priming experiments of Nikishin, Schwarzer, et al. (2022). This is orthogonal to a previously studied phenomenon where target network-based updates lead to perpetually reduced effective rank (Kumar et al. 2021).

3.2.2 On the potential causes of divergence

We conjecture that Q-value divergence starts with overestimated values of OOD actions. This overestimation could cause the optimizer to continually increase Q-values via its momentum leading to divergence. To test this hypothesis, we add a conservative behavioral

cloning (Pomerleau 1988; Atkeson and Schaal 1997) loss term to our actor that forces the policy to be close to replay buffer actions. Prior work employed this technique in offline RL to mitigate value overestimation (Fujimoto and Gu 2021). More formally, our actor update is extended by the loss $\mathcal{L}_{c,\psi} = \min_\psi \mathbb{E}_{a\sim\mathcal{D},\hat{a}\sim\pi_\psi(s)}[||a - \hat{a}||_2]$. The results in Figure 3.2 indicate that the basis of the conjecture is corroborated as divergence is much smaller—but not mitigated completely—when values are learned on actions similar to seen ones. However, in practice we do not know when divergence sets in, which limits the applicability of this technique in realistic scenarios. Using it throughout all of training, rather than just during priming, impairs the learner's ability to explore. We investigate the effects of the optimizer in more detail and provide preliminary evidence that the second-order term may be at fault in Appendix **??**.

3.2.3 Applying common regularization techniques

Regularization is a common way to mitigate gradient explosion and is often used to address overestimation (Farebrother, Machado, and Bowling 2018; Xinyue Chen, C. Wang, Zhou, and Keith W. Ross 2021; Liu et al. 2021; Hiraoka et al. 2022; Li et al. 2023). We investigate the priming experiments under techniques such as using L^2 weight decay (Krogh and Hertz 1991) or adding dropout (Srivastava et al. 2014) to our networks in Figure 3.3.

Both L^2 weight decay as well as dropout can somewhat reduce the divergence during priming, however not to a sufficient degree. While L^2 regularization fails to attain very high performance, dropout is able to recover a good amount of final return. However, both methods require tuning of a hyperparameter that trades off the regularization term with the main loss. This hyperparameter is environment-dependent and tuning it becomes infeasible for large UTD-ratios due to computational resource limitations. Still, the results imply that it is in fact possible to overcome the divergence in priming and continue to learn good policies.

3.2.4 Divergence in practice

One question that remains is whether we can find these divergence effects outside of the priming setup. We find that, while priming is an artificially constructed worst case, similar phenomena happen in regular training on standard benchmarks when increasing update ratios (see Figure 3.4). Further, the divergence is not limited to early stages of training as it happens at arbitrary points in time. We therefore conjecture that divergence is not a function of amount of experience but rather one of state-action space coverage. Note that the reported Q-values have been measured on the observed training data, not

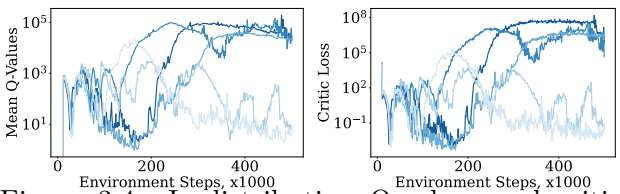

Figure 3.4: In-distribution Q-values and critic loss of five SAC seeds on the humanoid-run task using UTD = 32. Values diverge at arbitrary time-points, not only during the beginning. Loss

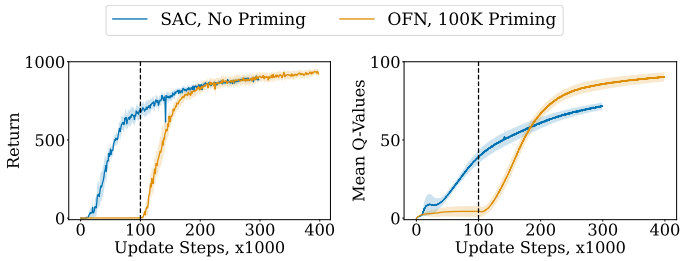

Figure 3.5: (Left) Return and (Right) Q-values comparing SGD result and OFN when priming for 100K steps. OFN obtains returns close to that of the well-trained SGD agent and learns an appropriate Q-value scale correctly.

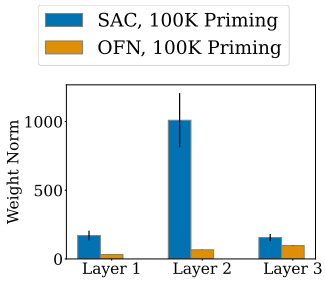

Figure 3.6: L_2 norm of network weights per layer after priming for default and OFN architectures. OFN leads to smaller weights and significant mass in the last layer.

on any out-of-distribution state-action pairs. The respective critic losses become very large. All this points toward a failure to fit Q-values. This behavior does not align with our common understanding of overfitting (Bishop 2006), challenging the hypothesis that high-UTD learning fails merely due to large validation error (Li et al. 2023).

3.3 Towards high-UTD optimization without resetting

Regularization techniques such as those in Section 3.2.3 can fail to alleviate divergence as they tend to operate across the whole network and lower the weights everywhere even if higher values are actually indicated by the data. They also require costly hyperparameter tuning. Thus, we turn towards network architecture changes to the commonly used MLPs that have proven useful in overcoming issues such as exploding gradients in other domains (J. L. Ba, Kiros, and Geoffrey E. Hinton 2016; Xu et al. 2019).

3.3.1 Limiting gradient explosion via unit ball normalization

As discussed previously, the prediction of an unknown action might trigger the propagation of a large, harmful gradient. Further, the Q-values of our network ought to grow over time as they more closely approximate those of a good policy. If we predict incorrectly on one of these Q-values, a potentially very large loss is propagated. Gradients are magnified by multiplicative backpropagation via ReLU activations (Glorot, Bordes, and Bengio 2011) as well as momentum from Adam (Kingma and J. Ba 2015). Note that all resulting issues arise

in the early network layers. We hypothesize that we can address many of these problems by separating the scaling of the Q-values to the appropriate size from the earlier non-linear layers of the network and moving the Q-value scaling to the final linear layer.

One contender to achieve the value decoupling described in the previous paragraph is layer normalization (J. L. Ba, Kiros, and Geoffrey E. Hinton 2016), but one would have to disable scaling factors used in common implementations. Still, standard layer normalization would not guarantee small features everywhere. Instead, we use a stronger constraint and project the output features of the critic encoder onto the unit ball using the function $f(\mathbf{x}) = \frac{\mathbf{x}}{\|\mathbf{x}\|_2}$ (B. Zhang and Sennrich 2019), where $\|\cdot\|_2$ denotes the L^2 vector norm and $\mathbf{x}$ is the output of our encoder $\phi(s, a)$. This ensures that all values are strictly between 0 and 1 and the gradients will be tangent to the unit sphere. Note that this function's gradient is not necessarily bounded to ensure low gradient propagation (see Appendix **??**), but we argue that if large values are never created in the early layers, gradient explosion will not occur. The unit ball has previously been used to mitigate large action prediction in the actor (C. Wang et al. 2020) or to stabilize RL training in general (Bjorck, Gomes, and Weinberger 2022). For brevity, we will refer to this method as *output feature normalization* (OFN). We solely apply OFN to the critic, unlike C. Wang et al. (2020), since our goal is to mitigate value divergence. OFN is very simple and requires only a one-line change in implementation.

3.3.2 Evaluating feature output normalization during priming

To test the efficacy of the OFN-based approach, we repeat the priming experiment in Figure 3.5. We find that OFN achieves high reward and most distinctly, Q-value divergence during priming is fully mitigated. Note also that we are using a discount factor of $\gamma = 0.99$, returns are collected over 1,000 timesteps and rewards are in $[0, 1]$. We therefore expect the average Q-values to be roughly at 10% of the undiscounted return which seems correct for the OFN network. However, more importantly, as shown in Figure 3.6, most of the Q-value scaling happens in the last layer.

3.4 Experimental evaluation

We evaluate our findings on the commonly used dm_control suite (Tunyasuvunakool et al. 2020a). All results are averaged over ten random seeds.[2] We report evaluation returns similar to Nikishin, Schwarzer, et al. (2022), which we record every 10,000 environment steps. We compare a standard two-layer MLP with ReLU (V. Nair and Geoffrey E Hinton 2010) activations, both with and without resetting, to the same MLP with OFN. The

[2]For comparison with TD-MPC2 (N. Hansen, Su, and X. Wang 2024) we use data provided by their implementation, which only contains three seeds. As the goal is not to rank algorithmic performance but to give intuition about the relative strengths of adapting the network architecture, we believe that this is sufficient in this case.

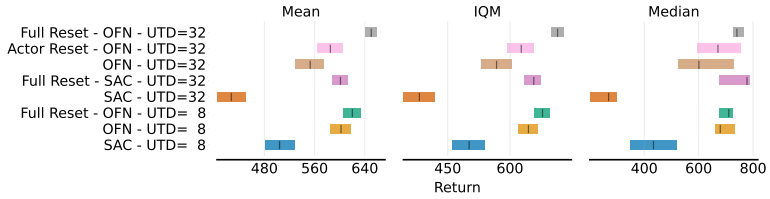

Figure 3.7: Mean, interquartile mean (IQM), and median with 95% bootstrapped confidence intervals of standard SAC and OFN on the DMC15-500k Suite. OFN can maintain high performance even under large UTD. OFN with UTD = 8 achieves comparable performance to standard resetting with UTD = 32 across metrics.

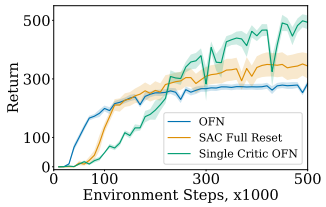

Figure 3.8: Mean return of single-critic OFN, standard OFN and resetting; UTD = 32 on hopper-hop. Shaded regions are standard error.

architecture is standard in many reference implementations. Architecture and the resetting protocol are taken from D'Oro, Schwarzer, et al. (2023) and hyperparameters are kept without new tuning to ensure comparability of the results. More details can be found in Appendix **??**.

To understand the efficacy of output normalization on real environments under high UTD ratios, we set out to answer multiple questions that will illuminate RL optimization failures:
Q1: Can we maintain learning without resetting neural networks?
Q2: Are there other failure modes beside Q-value divergence under high UTD ratios?
Q3: When resets alone fall short, can architectural changes enable better high-UTD training?

3.4.1 Feature normalization stabilizes high-UTD training

To answer **Q1**, we compare OFN and SAC with resets on the DMC15-500k benchmark with large update ratios of 8 and 32 as proposed by Nikishin, Schwarzer, et al. (2022) and Schwarzer, Obando Ceron, et al. (2023). We report mean, interquartile mean (IQM) and median as well as 95% bootstrapped confidence intervals aggregated over seeds and tasks, following Agarwal et al. (2021). The results are shown in Figure 3.7.

First, we observe that in both cases, UTD = 8 and UTD = 32, OFN can significantly improve over the non-resetting MLP baseline across all metrics. The value estimates that diverge seem to have been handeled properly (see Appendix **??**); learning is maintained. We note that our approach with UTD = 8 achieves mean and IQM performance comparable to that of standard resetting with UTD = 32. In median and quantile performance, all UTD = 32 overlap, highlighting that outliers contribute to the performance measurement. Note that the overall performance drops slightly for the OFN-based approach when going from UTD = 8 to UTD = 32. We conjecture that there are other learning problems such as exploration that have not been treated by alleviating value divergence. However, these do not lead to complete failure to learn but rather slightly slower convergence.

3.4.2 Other failure modes: Exploration limitations

To validate the hypothesis of other failures and answer **Q2**, we run two additional experiments. First, our current focus is on failures of the critic; our proposed mitigation does not address any further failures that might stem from the actor. We defer a more detailed analysis of actor failure cases to future work. Instead, we test the OFN-based architecture again and, for now, simply reset the actor to shed light on the existence of potential additional challenges. For comparison, we also run a second experiment in which we reset all learned parameters, including the critic.

The results in Figure 3.7 indicate that actor resetting can account for a meaningful portion of OFN's performance decline when going from UTD = 8 to UTD = 32. The actor-reset results are within variance of the full-resetting standard MLP baseline. Further, we observe that there is still some additional benefit to resetting the critic as well. This does not invalidate the premise of our hypothesis, value divergence might not be the *only* cause of problems in the high UTD case. We have provided significant evidence that it is a *major* contributor. Resetting both networks of OFN with UTD = 32 outperforms all other baselines on mean and IQM comparisons.

To explain the remaining efficacy of critic resets, we examine the hopper-hop environment where standard SAC with resets outperforms OFN. In RL with function approximation, one might not only encounter over- but also under-estimation (Wu et al. 2020; Q. Lan et al. 2020; Saglam et al. 2021). We believe that hopper is sensitive to pessimism, and periodically resetting the networks might partially and temporarily counteract the inherent pessimism of the dual critic setup.

To obtain evidence for this conjecture, we repeated some experiments with a single critic. As OFN handles divergence it might not require a minimization over two critics (Fujimoto, Hoof, and Meger 2018). We compare OFN using a single critic and 32 updates per environment step to standard SAC and OFN in Figure 3.8. With a single critic, OFN does not get stuck in a local minimum and outperforms full resetting. Note that this is only true in few environments, leading us to believe that the effects of high-update training are MDP-dependent. In some environments we observe unstable learning with a single critic, which highlights that the bias countered by the double critic optimization and the overestimation from optimization we study are likely orthogonal phenomena that both need to be addressed. Most likely, there is a difficult trade-off between optimization stability and encouraging sufficient exploration, which is an exciting avenue for future research.

3.4.3 Limit-testing feature normalization

To answer **Q3**, we move to a set of training environments that is considered exceptionally hard for model-free approaches, namely the dog tasks of the DMC suite. Standard SAC

Figure 3.9: Mean return on the dog DMC tasks, comparing OFN to SAC with resets and the model-based TD-MPC2. Shaded regions indicate standard error. OFN outperforms SAC with resets, which is unable to learn and OFN with UTD = 8 and resetting is competitive with TD-MPC2.

can generally not obtain any reasonable reward and, due to their complex dynamics, these tasks are often tackled using model-based approaches such as TD-MPC2 (N. Hansen, Su, and X. Wang 2024) with complicated update procedures and carefully tuned network architectures. We evaluate SAC and OFN on the dog tasks and compare against TD-MPC2 in Figure 3.9.

First, observe that resetting without OFN obtains no improvement over a random policy. However, OFN with UTD = 1 can already obtain very good performance across all tasks, indicating that a major problem for SAC in these high-dimensional tasks is value divergence. When increasing the update ratio to 8 and adding resetting, we improve the performance of the OFN agent even further and can match the reported results of the strong model-based TD-MPC2 baseline.

We have already seen that resetting can take care of multiple optimization failures. However, these experiments also indicate that resetting is not a panacea as it is only effective when the initially learned policy can obtain some reward before being overwritten. This seems intuitive since resetting to a policy that cannot gather any useful data should not help. These results highlight that the early training dynamics of RL are highly important when it comes to training on complex environments and fitting early data correctly and quickly is crucial for success.

This also opens up the question why resetting in the humanoid environments in the previous sections can yield success even though very little reward is observed. Besides greater divergence due to larger observation spaces in the dog MDPs, we suspect that this might be related to the complexity of exploration. The ability of a random policy to obtain non-trivial reward and information about the environment has been shown to be a crucial factor in explaining the success of DRL methods in discrete environments (Laidlaw, Russell, and Dragan 2023), and similar phenomena might be in effect here.

3.5 Related work

Our work closely examines previous work on the primacy bias and the related resetting technique (Anderson 1992; Nikishin, Schwarzer, et al. 2022; D'Oro, Schwarzer, et al. 2023; Schwarzer, Obando Ceron, et al. 2023). Going beyond, overestimation and feature learning

challenges are a widely studied phenomenon in the literature.

Combatting overestimation Overestimation in off-policy value function learning is a well-established problem in the RL literature that dates back far before the prime times of deep learning (Thrun and Schwartz 1993; Precup, Richard S. Sutton, and Dasgupta 2001). The effects of function approximation error and the effect on variance and bias have been studied (Pendrith and Ryan 1997; Mannor et al. 2007) as well. With the rise of deep learning, researchers have tried to address the overestimation bias via algorithmic interventions such as combining multiple Q-learning predictors to achieve underestimation (H. v. Hasselt 2010; H. v. Hasselt, Guez, and Silver 2016; Z. Zhang, Z. Pan, and Kochenderfer 2017; Q. Lan et al. 2020), using averages over previous Q-values for variance reduction (Anschel, Baram, and Shimkin 2017), or general error term correction (D. Lee, Defourny, and Powell 2013; Fox, Pakman, and Tishby 2016). In the context of actor-critic methods, the twinned critic minimization approach of Fujimoto, Hoof, and Meger (2018) has become a de-facto standard. Most of these approaches are not applicable or break down under very high update ratios. To regulate the overestimation-pessimism balance more carefully, several authors have attempted to use larger ensembles of independent Q-value estimates (K. Lee et al. 2021; Peer et al. 2021; Xinyue Chen, C. Wang, Zhou, and Keith W. Ross 2021; Hiraoka et al. 2022). Ensembling ideas were also combined with ideas from distributional RL (Marc G. Bellemare, Dabney, and Munos 2017) to combat overestimation (Kuznetsov et al. 2020). Instead of addressing the statistical bias in deep RL, our study focuses on the problems inherent to neural networks and gradient based optimization for value function estimation. Work from offline-to-online RL has demonstrated that standard layer normalization can bound value estimates during offline training and mitigate extrapolation while still allowing for exploration afterwards (Ball et al. 2023). Layer normalization has subsequently been used to achieve generally strong results in offline RL (Tarasov et al. 2023). Our work is also related to a recent contribution using batch-normalization for increased computational efficiency by Bhatt et al. (2024) who focus on decreasing update ratios rather than increasing them. A concurrent work by Nauman, Bortkiewicz, et al. (2024) provides a large scale study on different regularization techniques to combat overestimation. This work also demonstrates the efficacy of SAC on the dog tasks when properly regularized but it does not highlight the effects of Q-value divergence from exploding gradients as a key challenge for this set of environments.

Combating plasticity loss Another aspect of the primacy bias is the tendency of neural networks to lose their capacity for learning over time (Igl et al. 2021), sometimes termed *plasticity loss* (**lyle2021understanding**; Abbas, R. Zhao, et al. 2023). Recent work mitigates plasticity loss using feature rank maximization (Kumar et al. 2021), regularization (Lyle, Zheng, Nikishin, et al. 2023), or learning a second copied network (Nikishin, Oh, et al. 2024). Some of the loss stems from neurons falling dormant over time (Sokar et al. 2023).

A concurrent, closely related work by Lyle, Zheng, Khetarpal, et al. (2024) disentangles the causes for plasticity loss further. They use layer normalization to prevent some of these causes, which is closely related to our unit ball normalization. Our work differs in that we focus on the setting of high update ratios and use stronger constraints to mitigate value divergence rather than plasticity loss.

3.6 Conclusion and future work

By dissecting the effects underlying the primacy bias, we have identified a crucial challenge: *value divergence*. While the main focus in studying increased Q-value has been on the statistical bias inherent in off-policy sampling, we show that Q-value divergence can arise due to problems inherent to neural network optimization. This optimization-caused divergence can be mitigated using the unit-ball normalization approach, which shines on the dm_control benchmark with its simplicity and efficacy. With this result, we challenge the assumption that failure to learn in high-UTD settings primarily stems from *overfitting* early data by showing that combating value divergence is competitive with resetting networks. This offers a starting point towards explaining the challenges of high-UTD training in more detail and opens the path towards even more performant and sample efficient RL in the future.

However, as our other experiments show, mitigating value overestimation through optimization is not the only problem that plagues high-UTD learning. To clearly highlight these possible directions for future work, we provide an extensive discussion of open problems in Appendix **??**. Additional problems, such as *exploration failures* or *suboptimal feature learning*, can still exist and need to be resolved to unlock the full potential of high-UTD RL.

Chapter 4

Value-gradient Aware Model Learning

This chapter is based on Claas A Voelcker, Victor Liao, Animesh Garg, and Amir-massoud Farahmand (2022). "Value Gradient weighted Model-Based Reinforcement Learning". In: *International Conference on Learning Representations*.

4.1 Introduction

Model-based Reinforcement Learning (MBRL) is a sample-efficient approach to obtain a policy for a given control problem. It solves the control optimization into two interleaved stages: model learning and planning. In the *model learning* stage, an approximate model of the environment is learned which is then utilized in the *planning* stage to generate new experience without having to query the original environment. Often this process is repeated by continuously updating the model with new experience and replanning based on the updated model. MBRL is an enticing paradigm for scenarios in which samples of the true environment are difficult or expensive to obtain, such as computationally intensive simulators, real-world robots, or environments involving humans, since the model can be used to generalize the policy to unseen regions of the state space. The approach has received a lot of attention and significant progress has been made in the field (**Hafner2020Dream**; Richard S Sutton 1990; Deisenroth and Rasmussen 2011; Levine and Koltun 2013; Moerland et al. 2023; Schrittwieser et al. 2020).

One of the core problems of model-based policy learning methods, however, is that the accuracy of the model directly influences the quality of the learned policy or plan (Schneider 1997; Kearns and Singh 2002; S. Ross and Bagnell 2012; E. Talvitie 2017; Luo et al. 2019;

Janner et al. 2019a). Model errors tend to accumulate over time, and therefore long-term planning under approximate models can lead to suboptimal policies compared to the performance achievable with model-free approaches. This is especially prevalent in settings with complex dynamics, such as robotic environments with discontinuities, which can be hard to model with common function approximation methods. These model approximation errors are nearly impossible to avoid with current methods due to limits of function approximation in model and value learning algorithms, which cannot fully capture the full distribution over dynamics functions perfectly, and the use of finite datasets.

Hence, it is important that a model is accurate *where it counts* for the planning procedure, by modelling dimensions and data points that have a higher impact on the planning. But this objective is not captured in most current MBRL methods, which generally use maximum likelihood estimation (MLE) to learn a parametric model of the environment without involving information from the planning process. The misalignment between the *model learning* and *planning* stages of MBRL has recently received renewed interest and is now commonly termed the *objective mismatch* of reinforcement learning (Lambert et al. 2020), but the problems has been investigated in earlier works (Joseph et al. 2013). Several recent papers have investigated the objective mismatch (**AyoubJiaSzepesvariWang2020**; Abachi, Ghavamzadeh, and Farahmand 2020; A. Zhang et al. 2021; Grimm, André Barreto, et al. 2020; Grimm, André Barreto, et al. 2021; Nikishin, Abachi, et al. 2022), but currently theoretical investigation and understanding of possible approaches do not perform well when applied to complex deep learning based approaches (Lovatto et al. 2020) or the proposed approaches rely on heuristics which might not be applicable generally (S. Nair, Savarese, and Finn 2020).

Summary of Contributions. We present the *Value-Gradient Weighted MOdel Learning* (VaGraM) which rescales the mean squared error loss function with gradient information from the current value function estimate. We demonstrate the advantage of the VaGraM loss over previous approaches via the analysis of the optimization behavior of the Value-Aware Model Learning framework (Farahmand, André Barreto, and Nikovski 2017; Farahmand 2018) and form two hypotheses for the lack of empirical performance gain despite theoretical intuition: (a) the theory does not account for the optimization trajectory induced by the loss function and (b) it also does not address how to counter problems that arise when the state-space is yet insufficiently explored in early stages of the model training. Our experiments show, qualitatively and quantitatively, that the VaGraM loss impacts the resulting state and value prediction accuracy, and that it solves the optimization problems of previously published approaches. Beyond pedagogical domains, we show that VaGraM performs on par with a current state-of-the art MBRL algorithms in more complex continuous control domains, while improving robustness to irrelevant dimensions in the state-space and smaller model sizes.

4.2 Background

We consider the discounted MDP setting $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$ (**Puterman1994MarkovDP**), where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space of an agent, p is a transition probability kernel, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a scalar reward function, and γ denotes the reward discount factor. Following the standard setting of reinforcement learning, the goal is to obtain an agent which maximizes the reward function while interacting with the environment by taking actions after an optimal (potentially stochastic) policy π^* without knowledge of the true transition kernel.

We will concentrate on value function-based methods to solve the reinforcement learning problem. With these, the aim is to learn a function $V_\pi : \mathcal{S} \to \mathbb{R}$ which represent the (discounted) reward obtained in state s by following policy π from there: $V_\pi(s) = \mathbb{E}_{(s_0, a_0, \dots)} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s \right]$. It is also helpful to define an action-value function $Q(s, a) = r(s, a) + \gamma \int p(s'|s, a) V(s') ds'$. Many approaches (**dqn**; Watkins and Dayan 1992; Z. Wang et al. 2016; Haarnoja et al. 2018b) try to learn this function by minimizing the deviations of the value function approximation to a bootstrap target: $\min_\phi \mathbb{E} \left[(Q_\phi(s, a) - (r(s, a) + \gamma \int p(s'|s, a) V(s$ This equation forms the core motivation for our investigation of MBRL.

4.2.1 Model-based reinforcement learning

In the MBRL framework, an approximate model $\hat{p}$ is trained from data to represent the unknown transition function p. We will use the word 'model' to refer to the learned approximation and 'environment' to refer to the unknown MDP transition function.[1]

We concentrate on the Dyna algorithm (Richard S Sutton 1990) and specifically investigate the impact of model errors on the planning procedure. Dyna uses a dataset $\mathcal{D}$ of past experiences from the environment $\mathcal{D} = (s_i, a_i, r_i, s'_i)_{i=1}^N$. A parametric model $\hat{p}_\theta$ of the environment is learned by a maximum likelihood estimate using $\mathcal{D}$: $\theta^* = \arg\max_\theta \sum_{i=1}^N \log \hat{p}_\theta(s'_i, r_i | s_i, a_i)$. This model $\hat{p}_\theta$ is then used to sample new next states $s'_{\text{model}} \sim \hat{p}_\theta(\cdot|s, a)$ to obtain better coverage of the state-action space. The samples are used to train the value function and policy as if they were samples from the environment. It is also possible to learn deterministic models, which we will denote as f_θ for clarity.

4.2.2 Key Insight: Model mismatch problem

One of the main drawbacks of model-based reinforcement learning is the fact that model errors propagate and compound when the model is used for planning (Schneider 1997; Kearns and Singh 2002; E. Talvitie 2017). As a simple example, assume that a sample is

[1] We limit the discussion in this chapter to learning the model while assuming that the reward function is either known or learned by mean squared error minimization. In the empirical experiments, the reward function is learned using regression.

collected from a deterministic model and has an error ϵ. A value function based method will use the model sample to compute a biased bootstrap target $r(s,a) + \gamma V(s'+\epsilon)$.

The impact of the modelling error on the value function therefore depends on the size of the error and the local behavior of the value function. As an extreme example take a value function that only depends on a subset of all state observation dimensions. In this case, a large error in an irrelevant dimension has no consequence on the obtained policy, yet a maximum likelihood loss for the model cannot properly capture this behavior without prior handcrafted features.

We can motivate the use of MLE (such as the mean squared error for a Gaussian model with fixed variance) as a loss function by an upper bound: $\sup_{V \in \mathcal{F}} |\langle p - \hat{p}, V \rangle| \leq ||p - \hat{p}||_1 \sup_{V \in \mathcal{F}} ||V||_\infty \leq \sqrt{\mathrm{KL}(p||\hat{p})} \sup_{V \in \mathcal{F}} ||V||_\infty$ (Farahmand, André Barreto, and Nikovski 2017), but this bound is loose and does not account for the geometry of the problem's value function. In our example above a mean squared error would penalize deviations equally by their L_2 norm without accounting for the relevance of the dimensions.

4.2.3 Value-aware model learning

To address the model mismatch, Farahmand, André Barreto, and Nikovski 2017 proposed *Value-aware Model Learning* (VAML), a loss function that captures the impact the model errors have on the one-step value estimation accuracy. The core idea behind VAML is to penalize a model prediction by the resulting difference in a value function. Given a distribution over the state-action space μ and a value function V, it is possible to define a value-aware loss function $\mathcal{L}_V(\hat{p}, p, \mu)$:

$$\mathcal{L}_V(\hat{p}, p, \mu) = \int \mu(s,a) \left| \overbrace{\int p(s'|s,a)V(s')\mathrm{d}s'}^{\text{environment value estimate}} - \overbrace{\int \hat{p}(s'|s,a)V(s')\mathrm{d}s'}^{\text{model value estimate}} \right|^2 \mathrm{d}(s,a)$$

and its empirical approximation $\hat{\mathcal{L}}_V$ based on a dataset $D = (s,a,s')_{i=1}^N$ of samples from μ and p:

$$\hat{\mathcal{L}}_V(\hat{p}, \mathcal{D}) = \sum_{(s_i,a_i,s_i') \in \mathcal{D}} \left| V(s_i') - \int \left(\hat{p}(s'|s_i,a_i)\right) V(s')ds' \right|^2 .$$

It is worth noting that if the loss $\mathcal{L}_V$ is zero for a given model, environment and corresponding value function, then estimating the bootstrap target based on the model will result in the exact same update as if the environment were used. However, this is rarely the case in practice!

The main problem of this approach is that it relies on the value function, which is not

known a priori while learning the model. In the original formulation by Farahmand, André Barreto, and Nikovski (2017), the value function is replaced with the supremum over a function space. While this works well in the case of linear value function spaces, finding a supremum for a function space parameterized by complex function approximators like neural networks is difficult. Furthermore, the supremum formulation is conservative and does not account for the fact that knowledge about the value function is gained over the course of exploration and optimization in a MBRL approach.

Instead of the supremum over a value function class, Farahmand (2018) introduced a modification of VAML called *Iterative Value-Aware Model Learning* (IterVAML), where the supremum is replaced with the current estimate of the value function, . In each iteration, the value function is updated based on the model, and the model is trained using the loss function based on the last iteration's value function. The author presents error bounds for both steps of the iteration, but did not test the algorithm to ascertain whether the presented error bounds are sufficient to guarantee a strong algorithm in practice. Notably IterVAML provides an intuitive fix to the model-mismatch problem, yet overlooks two key optimization issues which lead to empirical ineffectiveness.

4.3 Value-Gradient Weighted MOdel Learning (VaGraM)

We present Value-Gradient Weighted MOdel Learning (VaGraM), a loss which is value-aware and has stable optimization behavior even in challenging domains with function approximation. To motivate the loss function, we highlight two causes for the lack of empirical improvements of IterVAML over MLE based approaches. These phenomena are investigated and verified in detail in Section 6.4.

Value function evaluation outside of the empirical state-action distribution IterVAML suffers when randomly initialized models predict next states that are far away from the current data distribution or if the optimization procedure leads the model's prediction outside of the covered state space. Since the value function has only been trained on the current data distribution, it will not have meaningful values at points outside of its training set. Nonetheless, these points can still achieve small value prediction errors if, due to the optimization process, the value function outside the training distribution happens to have the same value at the model prediction as at the environment sample. We therefore require that our value-aware loss function should not directly depend on the value function at the model prediction, since these might be potentially meaningless.

Suboptimal local minima Since the model can converge to a solution that is far away from the environment sample if the values are equal, we find that the model-based value prediction often performs poorly after updating the value function. We expect that the

updated model loss forces the model prediction to a new solution, but due to the non-convex nature of the VAML loss, the model can get stuck or even diverge. This is especially prevalent when the previous minimum is situated outside of the empirically covered state space. A stable value-aware loss function should therefore have only one minimum in the state space that lies within the empirical state distribution.[2]

4.3.1 Approximating a value-aware loss with the value function gradient

To derive a loss function that fulfils these requirements, we start from the assumption that the difference between the model prediction and the environment next states s' are small. This is implicitly required by many MBRL approaches, since an MLE model cannot be used to estimate the next state's value otherwise. We also assume that the model has small transition noise, akin to the model assumptions underlying MSE regression, otherwise the difference between a model sample and the next state sample might be large. Under this assumption, the IterVAML loss can be approximated by a Taylor expansion of the value function, where we denote the expansion of V around a reference point s' as $\hat{V}_{s'}$ and obtain $\hat{V}_{s'}(s) \approx V(s') + (\nabla_s V(s)|_{s'})^\intercal (s - s')$. Using this expansion at the next state sample $s'_i \in \mathcal{D}$ collected from the environment for each tuple independently instead of the original value function, the VAML error can be stated as:

$$\begin{aligned}\hat{\mathcal{L}}_{\hat{V}} &= \sum_{\{s_i, a_i, s'_i\} \in \mathcal{D}} \left(V(s'_i) - \int \hat{P}_\theta(s'|s_i, a_i)(V(s'_i) + (\nabla_s V(s)|_{s'_i})^\intercal (s' - s'_i))\mathrm{d}s' \right)^2 \\ &= \sum_{\{s_i, a_i, s'_i\} \in \mathcal{D}} \left(\int \hat{P}_\theta(s'|s_i, a_i) \left((\nabla_s V(s)|_{s'_i})^\intercal (s' - s'_i) \right) \mathrm{d}s' \right)^2\end{aligned}$$

This objective function crucially does not depend on the value function at unknown state samples, all s'_i are in the dataset the value function is trained on, which solves the first of our major problems with the VAML paradigm.

We can simplify the objective above even further if we restrict ourselves to deterministic models of the form $\hat{s}'_i = f_\theta(s, a)$. Since VAML requires the expectation of the value function under the model and the environment to be equal, we can exchange the probabilistic model with a deterministic one as long as we assume that the mean value function under the true environment is close to the empirical estimate of the value function from a single sample. We explore the prerequisites and consequences of this assumption further in **??**. The model

[2] The full loss function will likely still admit additional local minima due to the non-linear nature of the model itself, but the global optimum should coincide with the true model and the loss function should be convex in the state space.

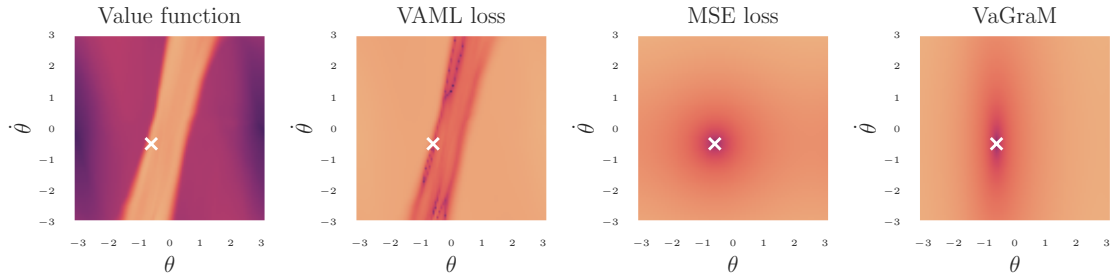

Figure 4.1: Visualization of discussed loss function with regards to a reference point marked with the white cross and the corresponding value function on the Pendulum environment. For the value function, darker color indicates a lower value. In the loss figures, darker color indicates how large the loss is if the model predicts $(\theta, \dot{\theta})$ instead of the reference sample marked in white. The VAML loss has a complex non-linear shape in the state space that follows isolines of the value function, while MSE and VaGraM are centered around the sample. For VaGraM, the rescaling of the MSE in the direction of high gradient along the θ axis is visible. Due to Figure 4.3.2, the scaling is aligned with the axis of the coordinate system and not rotated to fit the value function closer.

loss can then be expressed as:

$$\sum_i \left((\nabla_s V(s)|_{s'_i})^\intercal (f_\theta(s_i, a_i) - s'_i) \right)^2$$

We can see that the objective is similar to a mean squared error regression with a vector that defines the local geometry of the objective function. This vector can be interpreted as a measure of sensitivity of the value function at each data point and dimension. In regions where the value function changes significantly, the regression incentivizes the model to be very accurate.

4.3.2 Preventing spurious local minima

The formulation above retains one problem, Subsection 4.3.1 does not constrain the solution for each (s, a, s') tuple sufficiently. For each (s, a, s') tuple, the loss function only requires that the difference between the model and environment sample be orthogonal to the gradient of the value function, which describes a hyperplane of solutions. These predictions can lie arbitrarily far away from the environment sample, which breaks the assumption underlying the Taylor approximation that the model prediction is within a small region of the expanded state point. For more details see Section A.1.

To prevent these suboptimal solutions and achieve our second design goal, we consider an upper bound on the value-gradient loss by applying the Cauchy Schwartz inequality $(\sum_{i=1}^n x_i)^2 \leq n \sum x_i^2$ to change the square of the sum with a sum of squares. We denote the diagonal matrix with vector a on the diagonal as $\text{diag}(a)$ and refer to the dimensionality of

the state space as $\dim(\mathcal{S})$ and rephrase the sum as a vector-matrix multiplication:

$$\sum_{\{s_i,a_i,s_i'\}\in\mathcal{D}} \left((\nabla_s V(s)|_{s_i'})^\intercal (f_\theta(s_i,a_i) - s_i')\right)^2$$
$$\leq \dim(\mathcal{S}) \sum_{\{s_i,a_i,s_i'\}\in\mathcal{D}} \left((f_\theta(s_i,a_i) - s_i')^\intercal \mathrm{diag}(\nabla_s V(s)|_{s_i'})^2 (f_\theta(s_i,a_i) - s_i')\right).$$

This reformulation is equivalent to a mean squared error loss function with a per-sample diagonal scaling matrix. Because the scaling matrix is positive semi-definite by design, each summand in the loss is a quadratic function with a single solution as long as the derivative of the value function does not become zero in any component. Therefore this upper bound assures our second requirement: the loss function does not admit spurious local minima.[3]

To give an intuitive insight into all the discussed loss functions, we visualized each one for a pedagogical environment, the Pendulum stabilization task. The resulting loss curves can be seen in Figure 4.1. The VAML loss has a complicated shape that depends on the exact values of the value function while both MSE and our proposal have a paraboloid shape. Compared to MSE, our proposed loss function is rescaled to account for the larger gradient of the value function in the θ axis.

4.4 Experiment: Model learning in low-dimensional problem

We compare the performance of VaGraM, with both MSE and VAML on a pedagogical environment with a small state space and smooth dynamics to gain qualitative insight into the loss surfaces. We use the Pendulum environment, a canonical control problem in which an under-actuated pendulum must be swung and stabilized to an upright position. We use the implementation provided by Brockman et al. 2016. To learn the policy and its value function, we use the SAC algorithm (Haarnoja et al. 2018b). The original IterVAML analysis assumed that the value function was obtained using approximate value iteration (AVI) (Gordon 1995; Ernst, Geurts, and Wehenkel 2005; Farahmand, Szepesvári, and Munos 2010). We use SAC instead of a full AVI for stability in large scale experiments and discuss a proper extension of the VAML loss to SAC in **??**. We find that the difference in loss is negligible and therefore use SAC together with VAML throughout our experiments. More information on the implementation and hyperparameters of all of our experiments can be found in **??**.

[3] We note that state dimensions are ignored for points in which components of the value function become zero, potentially leading to additional solutions, but in practice this rarely happens for more than a few points.

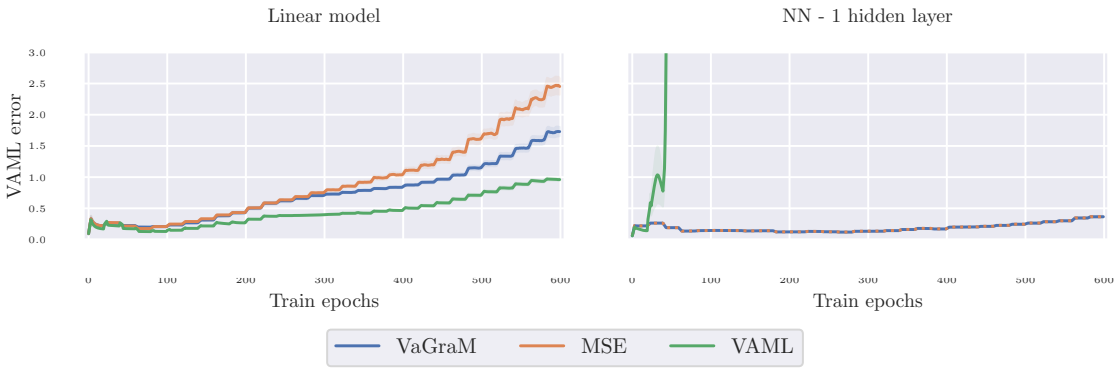

Figure 4.2: **Evolution of the VAML loss over changing value functions on the Pendulum domain**. Lines denote the mean and shaded areas show standard error over 8 model initialization and data set samples per model. In the linear setting, VAML achieves the lowest VAML error, while VaGraM is able to significantly outperform MSE. In the NN setting, VAML diverges rapidly, while VaGraM and MSE converge to approximately the same solution.

To simplify the setup of the evaluation, we decided to investigate the model losses without model-based value function learning. This allows us to focus solely on the loss functions, without taking into account the inter-dependency between model and value function updates. Instead of the model-based loop, we used the SAC algorithm in a model-free setup to estimate the value function. We saved the intermediate value functions after each epoch of training, corresponding to 200 environment steps, and optimized the models using stochastic gradient descent on the respective loss function, updating the value function used for the loss every 1000 model training steps. As the MLE loss, we used the mean squared error which assumes a Gaussian model with fixed variance.

To compare the optimization, we used two architectures, a linear regression without feature transformations and a neural network with a single hidden layer and 16 neurons. We obtained a dataset of states sampled uniformly over the whole state space and used the environment transition function to compute ground truth next state samples. Finally, we evaluated each models VAML error with regards to the current value function on a held out dataset and plotted the results in Figure 4.2.

Dependency on untrained value function estimates. The first cause for lacking empirical performance with VAML that we discussed in Subsection 7.4.1 was that the algorithm can predict successor states with incorrect value function as they lie outside of the data distribution.

In our experiment we find that a linear regression model remains stable under all three loss functions, VaGraM, MSE and VAML. But when using a flexible function approximation, the VAML loss converges in the first iteration with the given value function, but then rapidly

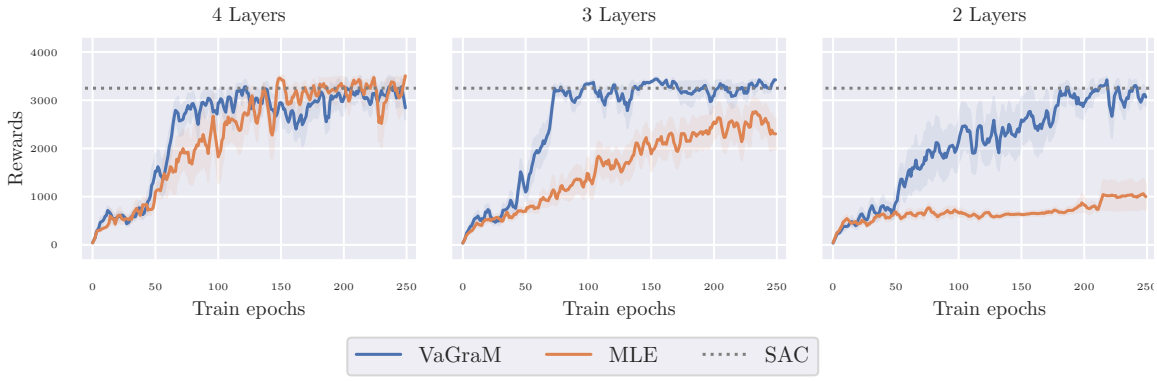

Figure 4.3: **Performance of VaGraM and MLE models with reduced model size**. The dotted lines correspond to the final performance reported for model-free SAC (grey, approx. 3200). Shaded area represents standard error over 16 repeated runs. VaGraM continues to solve the task almost unimpeded, while MLE is unable to even stabilize the Hopper when using a two layer neural network.

diverges once the value function is updated. When investigating the mean squared error of the VAML solution, we find that the model finds a stable minimum of the VAML loss outside of the reachable state space of the pendulum. This confirms our hypothesis that flexible VAML models can find solutions outside of the empirical state space distribution, which are unstable once we update the value function. VaGraM remains stable even with flexible function approximation and achieves a lower VAML error than the MSE baseline when using a model with insufficient capacity to represent the dynamics.

Single solution convergence. In the experiment, we see that the MSE and VaGraM models converge to a similar solution when using a neural network. This leads us to the conclusion that our loss function really only admits a single solution and that this solution coincides with the mean square error prediction when the function approximation has sufficient capacity to model the dynamics function with high precision. On the other hand, the VAML network converges to solutions that are far away from the environment sample measured in the L_2 norm and it cannot recover from these spurious minima due to the complex optimization landscape.

4.5 Experiment: Model-based Continuous Control

Due to the limited complexity of the pendulum environment, the quantitative differences between the mean squared error and VaGraM are at times insignificant in this setting. The dynamics function of the environment can be approximated sufficiently well with a simple neural network and one hidden layer.

The underlying theory supporting VAML states that a value-aware loss is preferable to a

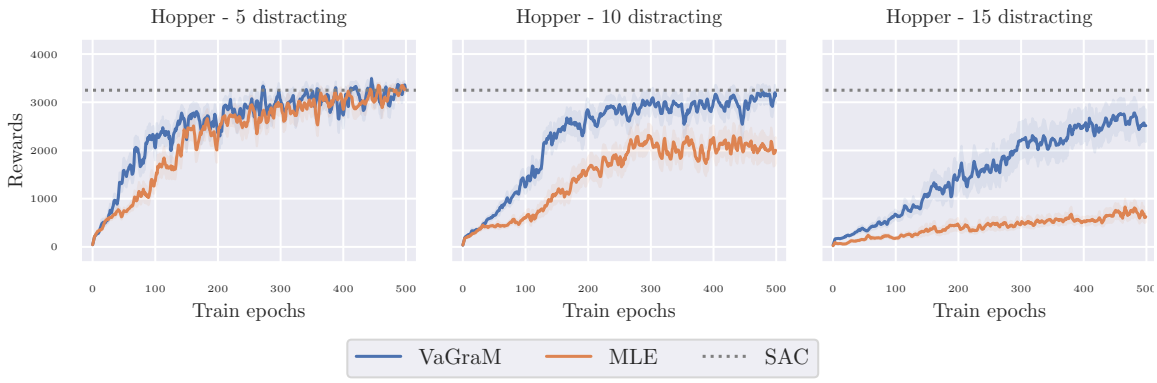

Figure 4.4: **Performance of VaGraM and MLE models with distracting state dimensions**. The dotted lines correspond to the final performance achieved by both algorithms on the Hopper task without distraction (grey, approx. 3200). Shaded area represents standard error over 16 repeated runs. VaGraM achieves significantly higher returns than the MLE baseline, especially in the most challenging setting with 15 distracting dimensions.

mean squared error loss in a setting where the capacity of the model is too small to fully approximate the problem or the state space contains dimensions that are irrelevant for the control problem. To test whether our loss function is superior to a maximum likelihood approach in these cases, we used the Hopper environment from the OpenAI gym benchmark (Brockman et al. 2016). As a deep learning based Dyna algorithm, we chose Model-based Policy Optimization (MBPO) (Janner et al. 2019a) and ran all of our experiments using the implementation provided by **Pineda2021MBRL**. We kept the structure of the MBPO algorithm and models and replaced the model loss function with VaGraM.

4.5.1 Hopper with reduced model capacity

In the first experiment, we decreased the network size of the used neural network ensemble. Janner et al. 2019a use fully connected neural networks with four hidden layers and 200 neurons per layer. To test the performance of the algorithm under smaller models, we ran tests with two and three layer networks and 64 neurons per hidden layer. The results are shown in Figure 4.3. As before, when using a sufficiently powerful function approximation, we see no difference between the maximum likelihood approach and VaGraM, suggesting that the networks are flexible enough to capture the true environment's dynamics sufficiently close for planning. But when reducing the model size, the maximum likelihood models quickly lose performance, completely failing to even stabilize the Hopper for a short period in the smallest setting, while VaGraM retains almost its original performance.

4.5.2 Hopper with distracting dimensions

To show that VaGraM is able to achieve good performance in a setting where there are additional dynamics in the environment that do not contribute to the control problem,

we appended distractor dimensions to the Hopper observations. These are independent of the original environment state space and reward function, and evolve under non-linear and discontinuous dynamics (details in **??**). A setting with distracting dimensions is known to pose difficulty for model-based control algorithms (**Stone2021TheDC**) and neural networks struggle to model non-linear, discontinuous dynamics, so with an increasing number of dimensions the task becomes harder.

The results of this experiment are shown in Figure 4.4. When using five distractor dimensions, both models are able to capture the dynamics sufficiently well to achieve comparable reward to the original environment. When increasing the number of dimensions, the performance of the MLE model deteriorates, as more and more of its capacity is used to model the added dynamics. VaGraM continues to be able to achieve reward even under the presence of distracting dimensions, since the gradient of the value function with regards to the state dimensions which are irrelevant to the control problem becomes small over training. Still, the performance of VaGraM also suffers with increasing dimensions: when adding 20 distracting dimensions, neither algorithm is able to stabilize the Hopper consistently. In this case, the value function approximation cannot differentiate sufficiently between the relevant and irrelevant dimensions with the amount of environment samples provided.

In summary, we find that VaGraM is able to deal with challenging distractions and reduced model capacity significantly better than a MLE baseline. This validates that our algorithm is really value-aware and can use the value function information to improve the performance of a model-based controller in settings where the model is unable to fully represent the environment.

4.6 Related work

Several authors have noted on the problem of learning models that align with the goal of obtaining a good policy. The proposed approaches fall into three broad categories: value-function or policy dependency, representation learning, and data resampling.

Inspired by VAML, Abachi, Ghavamzadeh, and Farahmand 2020 present a method that seeks to align the policy gradients under a learned model with the gradients under the true environment. Similar to our proposal, D'Oro, Metelli, et al. 2020 also proposed to reweigh samples in a log likelihood loss, but used policy search as the reinforcement learning approach and did not account for individual state dimensions. Nikishin, Abachi, et al. 2022 show an approach to directly optimizing the policy performance on the real environment by learning a model with implicit differentiation. Asadi et al. 2018 show that the original VAML loss coincides with a Wasserstein distance in the model space under the assumption that the value function is Lipschitz smooth. We find that all of these approaches suffer from similar scaling issues as VAML and have not been shown to lead to strong empirical

performance outside of toy settings.

Grimm, André Barreto, et al. 2020 characterize the space of possible solutions to the model learning problem under different restrictions given by value functions and policies, and propose to learn models that are value-equivalent, similar to Farahmand, André Barreto, and Nikovski 2017. In a follow-up work Grimm, André Barreto, et al. 2021 expand on this idea and show that their principle can be used to improve the MuZero algorithm (Schrittwieser et al. 2020). However, these works do not discuss the optimization challenges in actually finding such value-equivalent models, they mostly characterize the space under different value functions and policies, and present an orthogonal research direction to this chapter.

An alternative to characterizing the problem in the state space of the MDP are representation learning approaches, which seek to map the original states to a latent representation that is more amenable to control. Such approaches include Value Prediction Networks (**NIPS2017_ffbd6cbb**), Embed-to-Control (Watter et al. 2015) and related proposals (**Levine2020Prediction**; Cui, Chow, and Ghavamzadeh 2021). A. Zhang et al. 2021 build on the idea of bisimulation metrics (Ferns, Panangaden, and Precup 2004; Ferns, Panangaden, and Precup 2011) which seeks to characterize the difference between MDPs by finding a state mapping that is reward-invariant under policy and value function. In this work, we did not investigate learning state embeddings, but combining our work with representation learning approaches is an exciting direction for future research.

Lambert et al. 2020 hypothesize that the objective mismatch could be solved by reweighing the training buffer for model learning to prioritize datapoints with high value. These are more likely to matter for obtaining an optimal policy. A similar proposal was evaluated empirically by S. Nair, Savarese, and Finn 2020. Contrary to our work however, this technique cannot account for the differing impact of the state space dimensions and scaling, since the data points are weighted as a whole.

4.7 Conclusion

We presented the Value-Gradient Weighted MOdel Learning (VaGraM), a novel loss function to train models that model a dynamics function *where it matters* for the control problem. We derived our loss function from the value-aware model learning framework, showing that previous work does not account for two important optimization phenomena that appear when learning models with empirical value function approximations. We highlighted how VaGraM counters these issues and showed the increased stability of the training procedure when using our loss in a pedagogical environment.

On the Mujoco benchmark, VaGraM performs on par with maximum likelihood estimation

when using large neural network models. However, introducing additional complications to the problem results in drastic performance impacts for MLE based models, which highlights the necessity for value function aware losses in challenging environments and settings in which sufficient model capacity cannot be guaranteed. In these cases, value-awareness can greatly increase the performance of Dyna algorithms by focusing the model learning procedure on relevant aspects of the state space. In future work we seek to scale our loss function to image-based RL, where relevant state space dimensions can vary over a task due to shifting camera angles. Furthermore, we seek to derive a related value-aware approach for partially observable domains that can take the state inference problem into account.

Chapter 5

Understanding Auxiliary Tasks for Value Function Learning

This chapter is based on Claas A Voelcker, Tyler Kastner, Igor Gilitschenski, and Amir-massoud Farahmand (2024). "When does self-prediction help? Understanding Auxiliary Tasks in Reinforcement Learning". In: *Reinforcement Learning Conference*.

5.1 Introduction

Since the emergence of deep learning, techniques for deep supervised learning have been successfully incorporated into reinforcement learning (RL) agents (**dqn**; Lillicrap et al. 2016). However, the RL setting contains additional complications such as non-stationary optimization target and the reliance on bootstrapping. These hurdles generally add instability to the RL training process, and recent work has identified the *failure to learn good features* as a central problem in deep RL (Lyle, Rowland, and Dabney 2022; Nikishin, Schwarzer, et al. 2022; Kumar et al. 2021).

To mitigate this failure, one common approach is to add auxiliary tasks to the learning objective (Jaderberg et al. 2017). Popular examples include predicting next state observations (Jaderberg et al. 2016) and predicting functions of the next state (Schwarzer, Anand, et al. 2021; Ni et al. 2024). To understand the performance of these approaches, recent literature (Tang, Z. D. Guo, et al. 2022; Le Lan, Tu, Rowland, et al. 2023) considers the *learning dynamics* of auxiliary task learning in simple linear surrogate models (Saxe, McClelland, and Ganguli 2014). One hypothesis in the literature is that observation reconstruction should provide better features than latent self-prediction (Behzadian, Gharatappeh, and Petrik 2019; Tang, Z. D. Guo, et al. 2022). However, this causes a theory-practice gap as

empirical work has found that latent self-prediction outperforms observation reconstruction across many benchmarks (Schwarzer, Anand, et al. 2021; Ni et al. 2024).

To address this gap, we pose two questions: *(a) How do auxiliary losses behave when combined with a TD loss?* Tang, Z. D. Guo, et al. (2022), Le Lan, Tu, Oberman, et al. (2022), and Tang and Munos (2023) have studied the learning dynamics of auxiliary tasks alone, evaluating their performance without addressing the interaction between the auxiliary task and the main goal, to learn a (correct) value function. *(b) How can we describe the behavior of auxiliary losses in the presence of distractions (states and transition dynamics irrelevant for the reward) and observation functions (different ways to measure the underlying state)?* MDP structures like distractions have been hypothesized to lead to differing performance between different auxiliary tasks (Ni et al. 2024), but to our knowledge no theoretical study has been established.

In Section 5.3, we present a formalization of distractions and observation functions. We use the framework of *factored MDPs* (Boutilier, Dearden, and Goldszmidt 2000) with Kronecker products (Mahadevan 2009) to represent a common class of distractions. To model observation functions, we use *linear reparametrization* as a tractable way to go beyond one-hot representations.

In Section 5.4 and Section 5.5 we analyze the features learned with observation prediction and latent self-prediction alone and in combination with TD learning. We also show how these stationary features change with the introduction of distractions and observation functions. From this analysis we find that latent self-prediction is a strong *auxiliary task*, while observation prediction is a strong feature learning method when used *alone*. The differences are highlighted in Figure 5.1. This bridges one of the biggest gaps between previous analysis of learning dynamics and empirical results.

In Section 5.6 we test the predictions derived from our theoretical framework by evaluating feature learning losses in the MinAtar suite (Young and T. Tian 2019). [1] We design ablations that mirror both our formalization and previous approaches to test distraction robustness in empirical environments. The theory partially predicts the performance differences in the test suite, validating that the insights we obtain from the simple linear surrogate models used for analysis are useful for practitioners. However, we also find surprising deviations from our predictions on some environments, suggesting that there is need for additional research to fully bridge the theory-practice gap.

[1] All code for our experiments is available at `https://github.com/adaptive-agents-lab/understanding_auxiliary_tasks`.

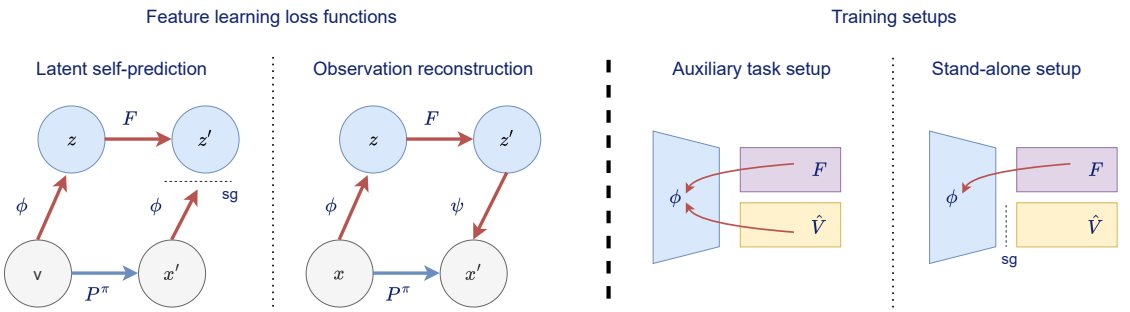

Figure 5.1: Diagram of the considered loss functions and the different use cases. In latent self-prediction, the aim is to predict next state *features* given by a embedding function $\phi(x')$ using the states features $\phi(x)$ and a latent prediction model F. In observation reconstruction, the aim is to match next state *ground truth observations* x' via the use of a decoder function $\psi(F(\phi(x)))$. In the *auxiliary task setup*, both the gradients from the feature learning loss and value function learning are propagated to the encoder, while in the *stand-alone scenario*, only the gradients from the feature learning loss are used to update Φ.

5.2 Background

We briefly introduce the standard formalism of reinforcement learning and linear value function approximations to clarify the notation used. Following this, we briefly introduce the training dynamics framework of Tang, Z. D. Guo, et al. (2022) and Le Lan, Tu, Rowland, et al. (2023). We frame all losses analyzed in this work with the same models to highlight similarities and differences.

Reinforcement Learning. We consider a discounted Markov decision process (MDP) (**Puterman1994Mark** Richard S. Sutton and Barto 2018a) $(\mathcal{X}, \mathcal{A}, \mathcal{P}, r, \gamma)$, with state space $\mathcal{X}$, action space $\mathcal{A}$, transition kernel $\mathcal{P} : \mathcal{X} \times \mathcal{A} \times \mathcal{X} \to \mathbb{R}$, reward function $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, and discount factor $\gamma \in [0, 1)$. Given a policy $\pi : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, the value function is defined as the expected return conditioned on a state x

$$V^\pi(x) = \mathbb{E}_\pi \left[\sum_{t \geq 0} \gamma^t r_t | x_0 = x \right],$$

where $r_t = r(x_t, a_t)$ is the reward at time t. The goal of an agent is to maximize its value at each state, using the (approximate) value function of its policy V^π.

In finite state spaces ($|\mathcal{X}| = n$ for some integer n), the policy induced transition kernel $\mathcal{P}^\pi(x'|x) = \int \mathcal{P}(x'|x, a) \mathrm{d}\pi(a|x)$ can be represented as a stochastic matrix P^π, and the reward function as a vector r^π. This permits the compact notation $V^\pi = (I - \gamma P^\pi)^{-1} r^\pi$. We review several additional properties of stochastic matrices in Subsection A.6.1.

Value function approximation. A finite state value function can always be expressed as a table or vector, but it is often infeasible to do so, due to limited storage capacity or resources

once the state space becomes sufficiently large. Therefore, function approximations need to be introduced, which commonly take the form of a feature function $\phi(x)$ and a weight vector $\hat{V}$, with $V^\pi(x) \approx \phi(x)^\top \hat{V}$. These features can be pre-specified or learned, i.e. using neural networks. Since we are interested in finite state spaces and linear models, we will assume that $\phi(x) : \mathbb{R}^n \to \mathbb{R}^k$ with $k < n$ represented by a matrix $\Phi \in \mathbb{R}^{n \times k}$ and $\hat{V} \in \mathbb{R}^k$.

5.2.1 Two-layer linear networks as analytical models for training dynamics

Rigorously analyzing the effect of different loss functions on neural networks is challenging due to non-linearities in the networks, shifting data distributions, and policy updates. Therefore, we have to resort to studying simplified models to obtain quantitative and qualitative results, and only consider the fixed policy case in our analysis[2]. Studying feature learning dynamics using linear networks was popularized by Saxe, McClelland, and Ganguli (2014) and has proven to be a valuable tool to analyze diverse objectives such as TD learning (Tang and Munos 2023; Le Lan, Tu, Rowland, et al. 2023), latent self-prediction (Y. Tian, Xinlei Chen, and Ganguli 2021; Tang, Z. D. Guo, et al. 2022), and linear autoencoders (Pretorius, Kroon, and Kamper 2018; Bao et al. 2020).

We rewrite the feature learning algorithms using two to three matrices in lieu of more complex functions. Furthermore, we use several assumptions throughout this chapter that are listed here for clarity.

Assumption 1. *Let* $\Phi \in \mathbb{R}^{n \times k}$ *be an* encoder *mapping to a* k *dimensional embedding space,* $F \in \mathbb{R}^{k \times k}$ *a* latent model *mapping to the next state's latent embedding,* $\hat{V} \in \mathbb{R}^k$ *and* $\hat{r} \in \mathbb{R}^k$ value and reward weights*, and* $\Psi \in \mathbb{R}^{k \times n}$ *a* decoder*. Let the sampling distribution of state samples* $\mathcal{D}$ *be uniform and fixed throughout learning.*

Using this notation, we study four important loss functions for RL: observation reconstruction, where the aim is to fit the next state observation $x^\top \Phi F \Psi \approx x'$, latent reconstruction $x^\top \Phi F \approx x' \Phi$, where the aim is to predict learned features of the next state, and TD learning $x^\top \Phi \hat{V} \approx x^\top (r^\pi + \gamma x'^\top \phi \hat{V})$. To clarify the differences, we show a diagram explaining the losses and training setups in Figure 5.1. Following common notation, $[\cdot]_{\text{sg}}$ signifies a *stop-gradient* operation; no gradient is taken with regard to terms in the parenthesis.

[2] We discuss these and other assumptions and their implications in detail in **??**.

Formally, these are written as

$$\begin{aligned}
\text{Reconstruction:} \qquad & L_{\text{rec}}(\Phi, F, \Psi) = \mathbb{E}_{x\sim\mathcal{D}}\left[\left\|x^\top \Phi F \Psi - x^\top P^\pi\right\|_2^2\right], \\
\text{Latent self-prediction:} \qquad & L_{\text{lat}}(\Phi, F) = \mathbb{E}_{x\sim\mathcal{D}}\left[\left\|x^\top \Phi F - \left[x^\top P^\pi \Phi\right]_{\text{sg}}\right\|_2^2\right], \\
\text{TD Learning:} \qquad & L_{\text{td}}(\Phi, \hat{V}) = \mathbb{E}_{x\sim\mathcal{D}}\left[\left\|x^\top \Phi \hat{V} - \left[x^\top \left(r^\pi + \gamma P^\pi \Phi \hat{V}\right)\right]_{\text{sg}}\right\|_2^2\right].
\end{aligned}$$

Following the nomenclature of Farahmand, André Barreto, and Nikovski (2017), we call the first two losses as *decision-agnostic*, and TD-learning as *decision-aware* as it depends on information specific to decision problem. To analyze the discrete-time learning dynamics, we study the analogous continuous-time gradient flow, which allows us to use the toolkit of dynamical systems theory. Writing α_Φ and α_F for the representation and model learning rates, i.e., the latent self-prediction dynamics are

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}\Phi_t &= -\alpha_\Phi \nabla_{\Phi_t} L_{\text{lat}}(\Phi_t, F_t) = -2\alpha_\Phi(\Phi_t F_t - P^\pi \Phi_t)F_t^\top, \\
\frac{\mathrm{d}}{\mathrm{d}t}F_t &= -\alpha_F \nabla_{F_t} L_{\text{lat}}(\Phi_t, F_t) = -2\alpha_F \Phi_t^\top(\Phi_t F_t - P^\pi \Phi_t).
\end{aligned}$$

We primarily consider the *two-timescale* regime, under the assumption that $\alpha_F \to \infty$ (Tang, Z. D. Guo, et al. 2022). Intuitively, this describes a learning setup in which the latent model is learned "much faster" than the latent mapping. This results in the following dynamics for self-predictive learning:

$$F_t^* = \left(\Phi_t^\top \Phi_t\right)^{-1} \Phi_t^\top P^\pi \Phi_t, \quad \frac{d}{dt}\Phi_t = \left(I - \Phi_t \left(\Phi_t^\top \Phi_t\right)^{-1} \Phi_t^\top\right) P^\pi \Phi_t F_t^{*\top}.$$

5.3 Formalizing the impact of distractions and observation functions

To bridge the theory-practice gap in feature learning, we formalize two structures found in MDPs found in Deep RL benchmarks that have, to the best of our knowledge, not appeared in work analyzing feature learning: observation functions and distractions. To allow a close comparison with previous work, our changes to the formalism are minimal on purpose, while still highlighting the important role these changes play in different loss functions.

5.3.1 Observation functions

Previous literature (Tang, Z. D. Guo, et al. 2022; Tang and Munos 2023; Le Lan, Tu, Oberman, et al. 2022) has eschewed the underlying observation of states in their analysis

of representation dynamics. The correctness of the dynamical system in Assumption 5.2.1 hinges on the fact that $\mathbb{E}[xx^\top] = I$, which implies uncorrelated state representations for each state x and a uniform distribution over states. The simplest form of such a representation would be a *one-hot* vector, a representation for the i-th state in which all entries are 0, except the i-th, which is 1.

This leads to an assumption that the features for the underlying states can be learnt independently. With a one-hot representations, the features of x are simply $x^\top\Phi = \Phi[i]$, the i-th row of Φ. A more realistic setting, which we focus on, is considering observation functions acting on the underlying states. This allows for representing systems where some states have correlated observations, which may be helpful *or* harmful for the RL problem. We provide a motivating example and a more extensive discussion regarding the effect observation functions on the learning process in Subsection A.5.1.

We briefly state this for reference later.

Assumption 2. *Let an* observation function *for a finite state space MDP be an invertible matrix* $\mathcal{O} \in \mathbb{R}^{n\times n}$.

Formalization: To introduce an observation function while remaining in the regime of analyzing linear networks and finite state problems, each one of n states is mapped to a unique $n-$dimensional observation vector by an invertible *observation matrix* $\mathcal{O} \in \mathbb{R}^{n\times n}$.[3] Invertibility is assumed to ensure the Markov property with linear function approximation.

This change from one-hot vectors to arbitrary vectors allows us to account for similarity. For example, if two states have almost identical observation vectors, they will be mapped to similar points in the latent representation space unless the features directly counteract this. We study the impact of changing the observations with a linear reparameterization in Subsection 5.4.2 by replacing x with $\bar{x} = \mathcal{O}^T x$.

5.3.2 Distracting state dynamics

In addition to observation functions, another common problem that many reinforcement learning algorithms face are distractions. While distractions have been a focus of empirical work studying the relative efficacy of different auxiliary tasks (Ni et al. 2024), a simple formalism whose effect on learning dynamics can be analyzed has not been presented. We propose to model an MDP with distractions using factored MDPs (Boutilier, Dearden, and Goldszmidt 2000).

Definition 11 (A factored MDP model of a distraction)**.** *Let* $M = (\mathcal{M}, P_M, R_M)$, $N =$

[3] We could also consider projection into higher dimensional spaces $\mathcal{O} \in \mathbb{R}^{n\times d}$ with $d > n$, without violating the Markov assumption, but this leads to additional complications (working with pseudo-inverses instead of inverses) which do not contribute meaningfully to the insights in this work.

$(\mathcal{N}, P_N, R_N)$ be a pair of Markov decision processes. The product process $M \otimes N$ is a MDP with state space $\mathcal{M} \times \mathcal{N}$, transition kernel $P_M \otimes P_N$ (where $\otimes$ signifies the Kronecker product), and reward function $R_M \otimes \mathbf{1} + \mathbf{1} \otimes R_N^\top$.

If $R_N = 0$, we refer to N as a distracting process, as it does not contribute to the reward.

This process models a common occurrence: two non-interacting processes unfold simultaneously, with the states being a combination of the two. Such a process can model a well-studied form of distraction, the background distractions in **Stone2021TheDC** or the random observation dimension in Nikishin, Abachi, et al. (2022) and Voelcker, Liao, et al. (2022). In this case, the foreground process M is assumed to carry the reward information, while the reward vector of the background process N is 0. We review important properties of the Kronecker product in Subsection A.6.1.

Note that our formalizations of observation functions and distracting processes is distinct from the assumptions in *linear MDPs* (Jin et al. 2020). Concretely we do not assume that the processes are low-rank compressible, just that they are factorizable.

5.4 Reconstruction and self-prediction losses

In this section and the next, we present an analysis of the stability conditions of reconstruction and self-prediction losses with linear networks. Using this analysis, we obtain several *insights*, qualitative predictions about how we expect the studied losses to behave in more complicated scenarios. These *insights* present the basis for our empirical comparison in Section 5.6.

5.4.1 Case 1: Orthogonal state representations

Tang, Z. D. Guo, et al. 2022 show that for symmetric MDPs, latent self-prediction converges to subspaces spanned by eigenvectors of P^π. We extend this result in the following sense: if P^π has positive real eigenvalues, invariant sub-spaces which are not spanned by the top-k eigenvectors are unstable for gradient descent.[4] It is interesting to note that the resulting features are identical to those obtained using the multi-reward approach described by Le Lan, Tu, Rowland, et al. 2023 (albeit under slightly different technical conditions), which highlights the close connection between the self-predictive approach and bootstrapped generalized value function learning. This furthermore suggests that using random rewards as auxiliary objectives (Farebrother, Greaves, et al. 2023) could result in very similar features as using self-prediction, which presentes an interesting avnue for further empirical study.

[4]The assumption of real *positive* eigenvalues is both more and less restrictive than the symmetry assumption made by Tang, Z. D. Guo, et al. (2022). An extension of our result to negative eigenvalues is presented in Section A.7, together with an extended comparison to the results obtained by Tang, Z. D. Guo, et al. (2022) and Le Lan, Tu, Rowland, et al. (2023).

Assumption 3. *Assume a two-timescale scenario and F_0 being initialized with full rank, and hence the non-collapse property ($\Phi_t^\top \Phi_t = \Phi_0^\top \Phi_0$) (Tang, Z. D. Guo, et al. 2022) holds.*

When referring to eigenvectors and singular vectors, we mean the *right* vectors of the corresponding matrices unless stated otherwise. We now present our first theoretical result.

Proposition 2 (Stationary points of latent self-prediction)**.** *Assume Assumption 1 and Assumption 2 hold. Furthermore, suppose P^π is real diagonalizable. If the columns of Φ_t span an invariant subspace of P^π, Φ_t is a stationary point of the dynamical system. Furthermore, if P^π is real-diagonalizable with positive eigenvalues, all invariant subspaces not spanned by the top-k eigenvectors sorted by eigenvalue are asymptotically unstable for gradient descent.*

This implies that even without the assumptions of symmetry of P^π required by Tang, Z. D. Guo, et al. (2022), the dynamics of latent self-prediction will tend to converge to invariant subspaces spanned by eigenvectors with large eigenvalues as other invariant subspaces are unstable. This is important as we expect these to be more important for representing potential reward functions in the environment (Le Lan, Tu, Rowland, et al. 2023).

We can contrast this with the features learned by a reconstruction loss. We write $\text{span}(A)$ for both the span of the column vectors of A or for the span of a set of vectors A, depending on context.

Proposition 3 (Stationary points of reconstruction)**.** *Assume Assumption 1 and Assumption 2 hold. Write $(u_1, \ldots, u_n)$, $(v_1, \ldots, v_n)$ for the left and right singular vectors of P^π sorted in descending order by singular value. Any stationary point (Φ^*, F^*, Ψ^*) of L_{rec} under the two timescale scenario satisfies* $\text{span}(\Phi^*) = \text{span}(\{u_1, \ldots, u_k\})$, $\text{span}(\Psi^{*\top}) = \text{span}(\{v_1, \ldots, v_k\})$.

Features of this form have been studied extensively and the convergence properties of linear auto-encoders are well understood (Baldi and Hornik 1989; Pretorius, Kroon, and Kamper 2018; Bao et al. 2020).

Proposition 8 and Proposition 9 together show that there is a subtle but important difference between latent self-prediction and observation reconstruction: the features will converge to eigenspaces in the former case, and to singular space in the latter case. Note that if P^π is a symmetric matrix, then the singular spaces and the eigen-spaces coincide and latent self-prediction and reconstruction converge to the same features (Tang, Z. D. Guo, et al. 2022).

Behzadian, Gharatappeh, and Petrik (2019) show that top k singular vectors are optimal low-rank linear features when making no assumptions on the reward, meaning observation reconstruction should lead to the best features when considering every possible bounded (or unknown) reward. Behzadian, Gharatappeh, and Petrik (2019) and Le Lan, Tu, Rowland,

et al. (2023) both highlight that if eigenvectors and singular vectors differ, singular vectors often lead to better performing features.

Insight 1 (Optimality of observation prediction)**.** *The features learned by observation prediction are in general superior to those of latent self-prediction, when using solely one of these as the loss function.*

5.4.2 Case 2: Observation function dependence

Recall that the gradient dynamics presented in (5.2.1) and analyzed in Proposition 8 and Proposition 9 rely on the assumption that $\mathbb{E}[xx^\top] = I$ (see Assumption 1). We now introduce the observation matrix $\mathcal{O}$, which leads to correlations between different features. To do this, we simply replace every occurence of $x^\top$ in the losses presented in Subsection 5.2.1 with $x^\top\mathcal{O}$. We assume that x is a one-hot vector as discussed before and the coverage is still uniform (Assumption 1 still holds), so all correlation between states arise as $\mathbb{E}[\mathcal{O}^\top xx^\top\mathcal{O}] = \mathcal{O}^\top\mathcal{O}$.

It is important to note that for BYOL and TD this rewriting leads to a linear basis change of Φ compared to the original loss, as each occurrence of $\mathcal{O}$ is multiplied by Φ. The only loss for which this is not the case is the reconstruction approach.

Proposition 4. *Assume Assumption 1, Assumption 2, and Assumption 3 hold. Let* $\{\Phi^*_{\text{lat/td}}\}$ *be the set of critical points of* L_{lat} *or* L_{td} *respectively. Then* $\mathcal{O}^{-1}\Phi^*_{\text{lat/td}}$ *are stationary points for the reparameterized losses* $L^{\mathcal{O}}_{\text{lat}}$ *and* $L^{\mathcal{O}}_{\text{td}}$.[5] *Furthermore, if* $\Phi^*_{\text{lat/td}}$ *is an asymptotically stable point of* $L_{\text{lat/td}}$ *that has a Jacobian with all negative eigenvalues,* $\mathcal{O}^{-1}\Phi^*_{\text{lat/td}}$ *is an asymptotically stable point of* $L^{\mathcal{O}}_{\text{lat/td}}$.

Note that while the stationary points and asymptotic stability conditions of the gradient flow might be unaffected by the introduction of observation distortions, the same might not be true for the dynamics of descent with finite step sizes. The numerical conditioning of the involved matrices change depending on $\mathcal{O}$ and so the impact of discretization due to finite step sizes changes the resulting dynamical system.

Proposition 5. *Assume Assumption 1, Assumption 2, and Assumption 3 hold. Let* $(u_1, \dots, u_n)$, $(v_1, \dots, v_n)$ *be the left and right singular vectors of* $\mathcal{O}^{-1}P^\pi\mathcal{O}$. *Any stationary point* (Φ^*, F^*, Ψ^*) *of* $L^{\mathcal{O}}_{rec}$ *satisfies* $\text{span}\,(\Phi^*) = \text{span}\,(\{u_1, \dots, u_k\})$, $\text{span}\,(\Psi^{*\top}) = \text{span}\,(\{v_1, \dots, v_k\})$.

The singular value decomposition of $\mathcal{O}^{-1}P^\pi\mathcal{O}$ will in general not have a clearly interpretable relationship to that of P^π and r, so the optimality result obtained by Behzadian, Gharatappeh, and Petrik (2019) do not hold in this case. However this does not mean that different observation functions will always harm the ability of the reconstruction loss to obtain good features. Consider for example an observation transformation that maps states directly

[5] Due to space constraints, we present the full equations in the proof.

to value and reward function. This would clearly be an example of a helpful observation transformation. However, in general we conjecture that arbitrarily changing the observation function will harm the reconstruction loss approach. A more detailed analysis involving the reward function of the problem being solved and its connections to the observation model are an exciting avenue for future work.

Insight 2 (Observation dependence of autoencoder models)**.** *Due to the invariance properties of latent self-prediction, we expect the performance of latent self-prediction to suffer less than the performance of observation reconstruction when perturbing the observation space arbitrarily.*

5.5 Understanding the effects on value function learning

The optimality of a representation for value estimation depends non-trivially on the structure of the reward structure of the MDP. Previous works (Behzadian, Gharatappeh, and Petrik 2019; Marc G Bellemare et al. 2019; Le Lan, Tu, Oberman, et al. 2022) attempted to reason about the optimality of representations without relying on the reward structure, by arguing that certain subspaces (such as the span of top-k eigenvectors or top-k singular vectors) are optimal given reward agnosticism.

Now we take a differing approach by taking the value function structure into account. Furthermore, we argue that the top-k eigenspaces (resp. singular spaces) are not always optimal. Indeed, we demonstrate in Appendix A.5.2 that with distractions, these subspaces can be particularly poor.

We begin by formalizing the reward function structure we will analyze. Let us write $w_1, \dots, w_n$ for the eigenvectors of P^π. We will assume that r^π has a low-dimensional structure in the following sense:

Assumption 4. *$\exists i_1, \dots, i_m \in \{1, \dots, |\mathcal{X}|\}$ such that $r^\pi \in \operatorname{span}(w_{i_1}, \dots, w_{i_m})$, and $m \leq k$. Let furthermore $\{w_{i1}, \dots, w_{im}\}$ be a minimal basis in the sense that $w_{in} {w_{in}}^\top r^\pi \neq 0$ for all n.*

We now write the summed losses

$$L_{\text{rec+td}}(\Phi, F, \psi, \bar{r}) = L_{\text{rec}}(\Phi, F, \psi) + L_{\text{td}}(\Phi, \bar{r})$$

and

$$L_{\text{lat+td}}(\Phi, F, \bar{r}) = L_{\text{lat}}(\Phi, F) + L_{\text{td}}(\Phi, \bar{r}).$$

Proposition 6. *Suppose that P^π is real diagonalizable, and that Assumption 1, Assumption 3, and Assumption 4 hold. There exists a non-trivial critical point Φ^* of the two-timescale TD loss L_{td} such that $\operatorname{span}(r^\pi) \subseteq \operatorname{span}(\Phi^*)$. Furthermore, Φ^* is a critical point*

of the two-timescale joint loss L_{td+lat}. Therefore combining L_{TD} and L_{Lat} does not exclude the existence of a stationary point with 0 value function approximation error.

We leave the extension of the stability result for TD learning to the joint loss case open for future work. Note that without the addition of TD learning, the latent loss would stabilize the top-k eigenspace representation, but we hypothesize that this behavior changes when combining the losses.

Proposition 7. *Let Assumption 1, Assumption 2, and Assumption 4 hold. If the reward spanning eigenvectors do not lie within the span of the top-k singular vectors,* $\text{span}\,(w_{i_1}, \ldots, w_{i_m}) \nsubseteq \text{span}\,(u_1, \ldots, u_k)$*, the critical points of the two-timescale joint loss L_{td+rec} are guaranteed to not be minimizers of the value function approximation error.*

Contrasting these propositions suggests that when combined with TD learning losses, latent self-prediction can be a more helpful auxiliary task. Indeed, when combining it with TD learning we can still guarantee that there exists an optimal combined solution. This does not hold for the reconstruction loss, where we can construct cases in which the joint loss leads to worse TD error.

Insight 3 (Latent self-prediction as an auxiliary task)**.** *For good performance across a wide variety of tasks, latent self-prediction needs to be combined with TD learning as an auxiliary task. It is a preferable auxiliary task to observation prediction in most scenarios, but especially in scenarios with distracting processes.*

5.6 Empirical study of theoretical results in deep learning based settings

We aim to empirically verify the statements marked as *"Insights"* throughout this chapter: superiority of observation prediction as a standalone feature learning loss (Insight 1), impact of the observation function on the different loss functions (Insight 2), and the relative strength of latent-self prediction as an auxiliary loss compared to reconstruction (Insight 3). As our theory addresses the simplified setting of policy evaluation with linear models, we seek to test if the insights transfer to the more common setting of control with neural networks. Across all experiments, we report mean performance over 30 seeds and shaded 95 bootstrapped confidence interval.

To test these hypotheses, we use the MinAtar suite of five Atari inspired videogames (Young and T. Tian 2019) and the DMC 15 suite (Tunyasuvunakool et al. 2020b).[6] Both are small enough to perform thorough investigations, while providing non-trivial observation spaces and dynamics. Detailed information about the implementation and hyperparameters can be found in **??**.

[6]DMC experiments are presented in the appendix due to space constraints.

Auxiliary task learning vs general purpose feature learning (Figure 5.2 and Figure 5.3): First, we compare both the auxiliary task and stand-alone feature learning scenarios. As expected from prior work (Jaderberg et al. 2016; Schwarzer, Anand, et al. 2021; Farebrother, Greaves, et al. 2023), in all cases using an auxiliary loss performs no worse (and often better) than vanilla DQN. We find that as expected from Insight 3, latent self-prediction is a stronger auxiliary loss function than observation reconstruction in three out of five environments. However, when using the decision-agnostic losses alone, we clearly see observation reconstruction performing significantly better than latent self-prediction, which fails to learn any relevant features in several cases. This verifies Insight 1.

Curiously, in the case of the Seaquest environment, we find that using observation prediction alone outperforms using it as an auxilliary task strongly, and performs on par with the auxiliary task variant of the latent self prediction loss. Seaquest also has the sparsest reward structure in the test suite, which can make it a challenging environment for DQN. In this case, features based purely on the observed transition might allow for better policy learning.

This highlights that no algorithm is clearly superior in all settings and the reward and observation structure is very relevant for the performance of each loss.

Observation space distortions (Figure 5.4): To test the impact of changing the observation function, we sample a random binary matrix and multiply it to the flattened observation vector. We then reshape the observation to the original shape again.

All algorithms show themselves to be strongly impacted by this random observation distortions, which suggests that our claim of invariance of self-prediction relies too strongly on the linear gradient-flow limit. This can in part be explained by the use of a convolutional layer in the standard baseline implementation of DQN which we adapted. However, we still find that at least on two environments (Seaquest and Freeway), the latent self-prediction auxiliary task is able to recover more of the original performance than either observation prediction or DQN. Interestingly, the DQN baseline seems to suffer the most from the introduction of

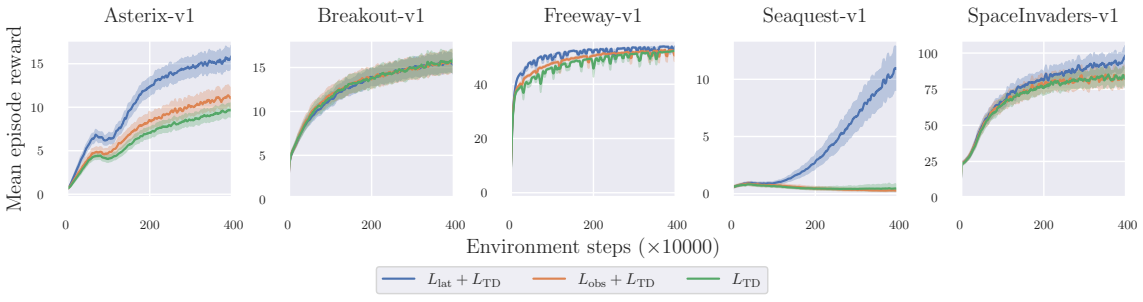

Figure 5.2: Auxiliary task setup: Performance of all losses on the observation space as given without changes to the environment.

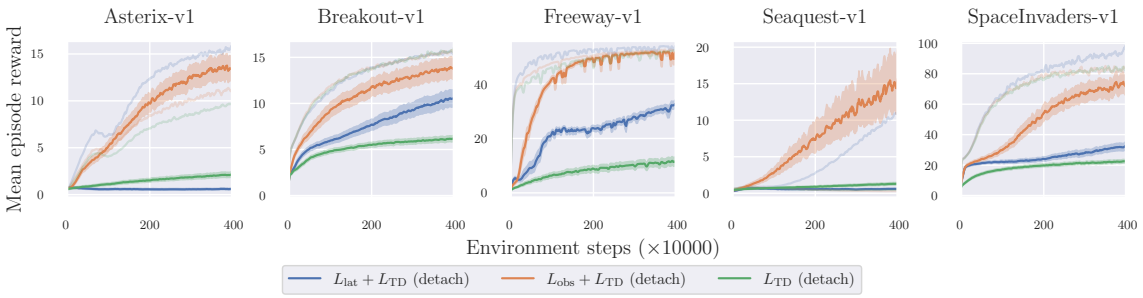

Figure 5.3: Stand-alone setup: Performance of all losses on the observation space as given without changes to the environment. The DQN baseline is using random features, which are not updated, to verify that learning features is indeed superior to a random feature baseline.

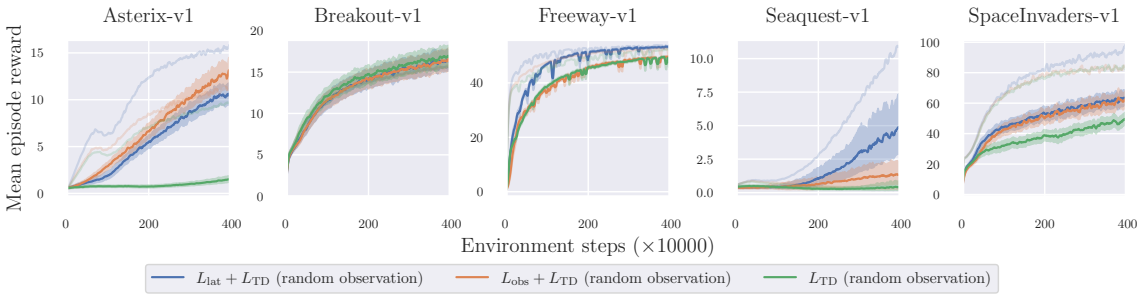

Figure 5.4: Distorted observation function with a random transformation.

the observation change, which suggests that correlations of the existing observation space play an important role in learning correct value function prediction.

Overall, we find that Insight 2 does not fully translate to the more complex test setting. In part, this may be explained by the fact that the original observation spaces of the test environments already violate our assumptions for the one-hot encoding. In addition, introducing linear correlation might not impact non-linear model learning in the same way it would impact linear models. This highlights the need for more in-depth research on the interplay between given observation space and feature learning.

Distractions (Figure 5.5 and Figure 5.6): As our results are dependent on the spectral structure of the environment, different distraction models can be assumed to have differing impact on the efficacy of the tested losses. This behavior is dependent on the structure of the noise. If the distraction does not strongly change the top-k singular or eigenspaces, it will be less problematic for the auxiliary tasks, especially for observation reconstruction. Testing the impact of different distraction models on the top-k spaces is out of scope for this work, but we conjecture that fully random noise has less structure than distractions following clear patterns.

Therefore, we consider two simple distraction models in our experiments. The distractions

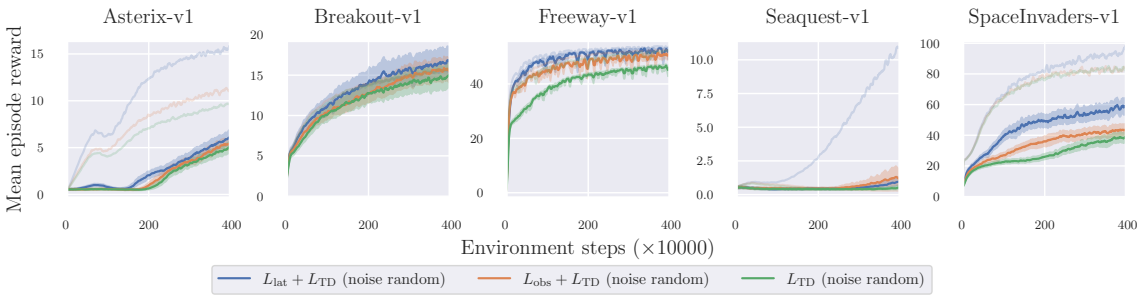

Figure 5.5: Appending random noise channels.

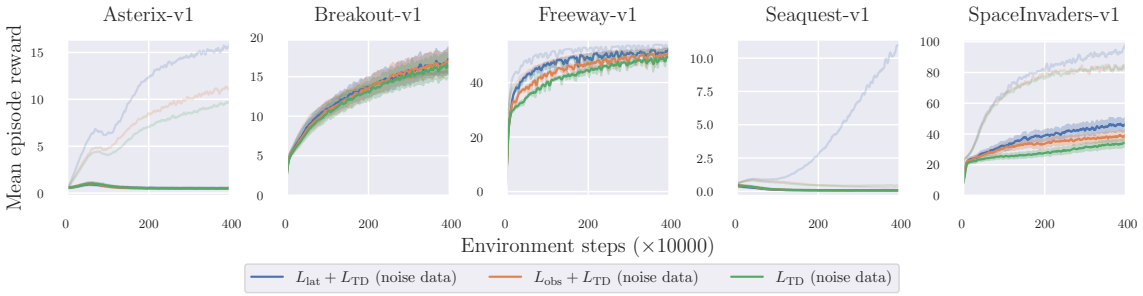

Figure 5.6: Appending structured noise channels using the Freeway environment.

are concatenated to the original observations along the channel dimension. First, we use random noise sampled independently for each state from a Bernoulli distribution. As there is no predictable structure in this noise, we expect all algorithms to be able to deal with this distraction better. Second, we choose one of the environments at random (Freeway-v1) and concatenate two copies to the observation space of each environment. The dynamics are obtained by sampling a random action at each timestep independent from the policy and stepping the distraction environment with it.

We find a small advantage in some environments to using the latent self-prediction loss and using random noise, and no clear advantage from any algorithm in the structured noise case. Structured noise poses a much larger challenge to most algorithms, completely preventing learning in several cases. This partially validates that not only the presence or absence of noise matters, but also how it changes relevant quantities, e.g. the eigenvalues of the transition kernel.

In the continuous control experiments presented in **??**, we find that the self-prediction loss performs generally better than in the MinAtar suite. As the observation and reward structure differs between these two benchmark suite, this obvservation lends more credibility to our claims that the observation structure impacts the performance of algorithms.

5.7 Conclusions

When choosing an RL approach to use, practitioners are overwhelmed with a variety of loss function choices, without clear indication which one will be preferable in what scenario. In our work, we introduce analytically tractable notions of distractions and observation functions. With these we predict the performance of latent self-prediction and observation reconstruction as stand-along feature learning methods and auxiliary tasks, and study our theoretical insights empirically. Our evaluation lends credibility to the use of simple surrogate models to obtain practically relevant insights into algorithmic performance. However, in several cases we also find deviations between our predictions and more complex benchmarks. Therefore, while we claim that our experiments have the ability to guide the choice of algorithms for applied settings, there is still a sizeable gap between theory and practice that remains to be bridged in future work.

We also note that our experiments showed substantial differences in behavior of auxiliary losses both within and across benchmarks and different noise distractions. Previous work that studied the effect of distraction (Nikishin, Abachi, et al. 2022; Voelcker, Liao, et al. 2022; Ni et al. 2024) did not discuss their distraction model in further details. In light of our results, we suggest that empirical research should be careful about the choice of benchmark and experimental setup and discuss the implications of the empirical setup explicitely.

One of the most important gaps between the work presented in this chapter and the behavior of online algorithms is the restrictive assumption of the fixed policy evaluation case. Therefore one of the most exciting avenues for future work is analysing policy improvement, where the underlying dynamics of the environment change due to the policy updates. On the empirical side the surprising effectiveness of the observation prediction loss on the Seaquest environment highlights the fact that even within benchmark suites, differences in the reward functions and observation models can lead to differing rankings between algorithms. This further highlights the necessity of studying the structure of MDPs and to design algorithms that are robust to different structures, or adapt to them automatically.

Chapter 6

Calibrated Latent Models for Value Aware Model Learning

This chapter is based on Claas A Voelcker, Anastasiia Pedan, Arash Ahmadian, Romina Abachi, Igor Gilitschenski, and Amir-massoud Farahmand (2025). "Callibrated value-aware loss functions with stochastic environment models". In: *under review* and Claas A Voelcker, Arash Ahmadian, Romina Abachi, Igor Gilitschenski, and Amir-massoud Farahmand (2024). *λ-models: Effective Decision-Aware Reinforcement Learning with Latent Models*. arXiv: 2306.17366 [cs.LG]. URL: https://arxiv.org/abs/2306.17366.

6.1 Introduction

In model-based reinforcement learning, an agent collects information in an environment and uses it to learn a model of the world to accelerate and improve value estimation or the agent's policy (**Hafner2020Dream**; Richard S Sutton 1990; Deisenroth and Rasmussen 2011; Schrittwieser et al. 2020). However, as environment complexity increases, learning a model becomes more and more challenging. These model errors can impact the learned policy (Schneider 1997; Kearns and Singh 2002; E. Talvitie 2017; Lambert et al. 2020) leading to worse performance. In many realistic scenarios, agents are faced with a "big world" where learning to accurately simulate the environment is infeasible. When faced with complex environments, deciding what aspects of the environment to model is crucial. Otherwise, capacity would be spent on irrelevant aspects, e.g., modelling clouds in a vehicle's video feed.

Recently, the paradigm of *decision-aware model learning* (DAML) (Farahmand, André Barreto, and Nikovski 2017) or *value equivalence* (Grimm, André Barreto, et al. 2020; Grimm,

André Barreto, et al. 2021) has been proposed. The core idea is that a model should make predictions that result in correct value estimation. Two prominent algorithms in the space of decision-aware model learning are the IterVAML (Farahmand 2018) and MuZero (Schrittwieser et al. 2020) algorithms. While IterVAML is a theoretically grounded algorithm, difficulties in adapting the loss for empirical implementations have been highlighted in the literature (Lovatto et al. 2020; Voelcker, Liao, et al. 2022). MuZero on the other hand has been shown to perform well in discrete control tasks (Schrittwieser et al. 2020; Ye et al. 2021) but has received little theoretical investigation. Thus, understanding the role of different value-aware losses and determining the factors for achieving strong performance is an open research problem.

In this work, we address this problem by establishing similarities and differences between these two approaches. Our goal is to establish an algorithmic framework and to provide insight into the strengths and weaknesses of each. Experimental evaluations focus on the state-based DMC environments (Tunyasuvunakool et al. 2020b), as the focus of this chapter is not to scale up to image-based observations.

The main differences between previous works trying to build on the IterVAML and MuZero algorithms are neural network architecture design and the value function learning scheme. MuZero is built on value prediction networks (Oh, Singh, and H. Lee 2017), which explicitly incorporate a latent world model, while IterVAML is designed for arbitrary model choices. We show that the use of latent models can explain many previously reported performance differences between MuZero and IterVAML (Voelcker, Liao, et al. 2022; Lovatto et al. 2020).

As a theoretical contribution, we prove a conjecture by Oh, Singh, and H. Lee (2017) that IterVAML and MuZero-based models can achieve low error in stochastic environments, even when using deterministic world models. To the best of our knowledge, we are the first to prove the conjecture. The MuZero value function learning scheme however results in a bias in stochastic environments. We show that this bias leads to a quantifiable difference in performance. Finally, we show that the model learning component of MuZero is similar to VAML, which was previously also noted by Grimm, André Barreto, et al. (2021).

Finally, we show when decision-aware losses provide benefits in practice and propose new ways to combine them with policy gradient estimation approaches. To highlight that many design decisions can be combined freely with one another, we propose to treat previous approaches as parts of a family of algorithms, which we call Latent Model-Based Decision-Aware Actor-Critic (λ-AC) methods. λ-AC algorithms combine three core components: a latent world model, a value-aware model loss function, and a model-based actor-critic algorithm for reinforcement learning.

In summary, our contributions are a) showing that IterVAML is a stable loss when a latent

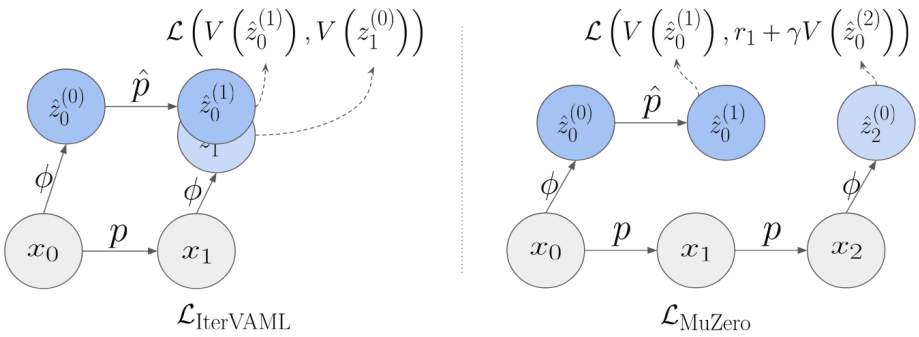

Figure 6.1: Sketch of the different value-aware losses with latent models. IterVAML computes the value function difference between the latent prediction and the next state encoding, while MuZero computes a single-step bootstrap estimate.

model is used, b) proving and verifying that the MuZero value loss is biased in stochastic environments, and c) showing how novel combinations of established algorithms provide benefits in benchmarks.

6.1.1 Decision-Aware Model Losses

The losses of the decision-aware learning framework share the goal of finding models that provide good value function estimates. Instead of simply learning a model using maximum likelihood estimation, the value function is directly used in the model update.

Given a distribution μ over the state space and a value function approximation $\hat{V}$, the IterVAML loss (Farahmand 2018) is defined as

$$\mathcal{L}_{\text{IterVAML}}(\hat{p}; \mu, \hat{V}, p) = \mathbb{E}_{x,a\sim\mu}\left[\left|\int_{\mathcal{X}} \left(\hat{p}(x'|x,a) - p(x'|x,a)\right) \hat{V}(x')\mathrm{d}x'\right|^2\right].$$

Although not originally presented by Farahmand 2018, this formulation can be easily extended to a multi-step version $\mathcal{L}^n_{\text{IterVAML}}$[1]. In practice, we assume access to a dataset of state and reward sequences $\mathcal{D} = \{x_{i_1}, r_{i_1}, \ldots, x_{i_n}, r_{i_n}\}_{i=1}^N$ with x_{i_1} sampled from μ and subsequent states sampled according to the transition function. Note that the index 1 here denotes the start of a sequence, not the start of an episode, and sequences can be overlapping. Then, the sample-based IterVAML loss is

$$\hat{\mathcal{L}}^n_{\text{IterVAML}}(\hat{p}; \hat{V}, \mathcal{D}) = \frac{1}{N \cdot n} \sum_{i=1}^{N} \sum_{j=1}^{n} \left[\mathbb{E}_{\hat{x}^{(j)}\sim\hat{p}^j(\cdot|x_{i_1},a_{i_1})}\left[\hat{V}\left(\hat{x}^{(j)}\right)\right] - \hat{V}\left(x_{i_j}\right)\right]^2.$$

The population-based MuZero loss is not defined in the literature, but the sample-based version is. Similar to **dqn**, it uses a target network for the value function, which we denote

[1] A more detailed discussion of this extension is found in Subsection A.3.5

as V_{target}. The MuZero loss is given as

$$\hat{\mathcal{L}}^n_{\text{MuZero}}\left(\hat{f}, \hat{V}; \mathcal{D}, V_{\text{target}}\right) = \frac{1}{N \cdot n} \sum_{i=1}^{N} \sum_{j=i}^{n} \left[\hat{V}\left(\hat{f}^j(x_{i_1}, a_{i_1})\right) - \left[r_{i_j} + \gamma V_{\text{target}}\left(x_{i_{j+1}}\right)\right]\right]^2.$$

Note that the real environment reward and next states are used in the j-th step. The main difference between the two loss formulations is that IterVAML uses a squared minimization between two value function outputs, while MuZero computes the value function target via bootstrap. For the 1-step case, this is illustrated in Figure 6.1. The MuZero loss is also used for updating both the value function and the model, while IterVAML is only designed to update the model and uses a model-based bootstrap target to update the value function.

Decision-aware losses can be insufficient for sample-efficient learning (Ye et al. 2021), especially in continuous state-action spaces (N. Hansen, X. Wang, and Su 2022). Their performance is improved greatly by using an auxiliary latent self-prediction loss based on the *Bootstrap your own latent* (BYOL) loss (Grill et al. 2020). Several different versions of this loss have been proposed for RL (Gelada et al. 2019; Schwarzer, Anand, et al. 2021; Tang, Z. D. Guo, et al. 2022), we consider a simple n-step variant:

$$\hat{\mathcal{L}}^n_{\text{latent}}\left(\hat{f}, \phi; \mathcal{D}\right) = \frac{1}{N \cdot n} \sum_{i=1}^{N} \sum_{j=1}^{n} \left[\hat{f}^{(j)}(\phi(x_{i_1}), a_{i_1}) - \text{stop-grad}\left[\phi(x_{i_j})\right]\right]^2.$$

Theoretical analysis of this loss by Gelada et al. (2019) and Tang, Z. D. Guo, et al. (2022) also shows that the learned representations provide a good basis for value function learning.

6.1.2 Actor-critic learning

Both IterVAML and MuZero can use the model to approximate the value function target. In its original formulation, MuZero is used with a Monte Carlo Tree Search procedure to expand the model which is not directly applicable in continuous state spaces. A simple alternative for obtaining a value function estimate from the model in continuous state spaces is known as model value expansion (MVE) (Feinberg et al. 2018). In MVE, short horizon roll-outs of model predictions $[\hat{x}^0, \ldots, \hat{x}^n]$ are computed using the current policy estimate, with a real sample from the environment as a starting point $x^{(0)}$. Using these, n-step value function targets are computed $Q^j_{\text{target}}\left(\hat{x}^{(j)}, \pi\left(\hat{x}^{(j)}\right)\right) = \sum_{i=j}^{n-1} \gamma^{i-j} \hat{r}\left(\hat{x}^{(i+j)}, \pi\left(\hat{x}^{(i+j)}\right)\right) + \gamma^{n-j} \bar{Q}\left(\hat{x}^{(n)}, \pi\left(\hat{x}^{(n)}\right)\right)$. These targets represent a bootstrap estimate of the value function starting from the j-th step of the model rollout. For our IterVAML experiments, we used these targets together with TD3 (**td3**) to learn the value function, for MuZero, we used these targets for V_{target} in Figure 6.1.1.

In continuous control tasks, estimating a policy is a crucial step for effective learning algorithms. To improve policy gradient estimation with a model, model-based value gradients (Heess et al. 2015; Amos et al. 2021) can be used. Model-based policy gradients (Heess et al. 2015) are obtained by differentiating the full n-step MVE target (Subsection 6.1.2) with regard to the policy. If differentiable policies and models are used, a policy gradient over the complete computational graph of the model rollout is obtained. We evaluate the usefulness of model-based policy gradients in Section 6.4, with deterministic policy gradients (Silver, Lever, et al. 2014; Lillicrap et al. 2016) as a model-free baseline.

6.2 Latent Decision Aware Models

To compare previous work in decision-aware MBRL (Farahmand, André Barreto, and Nikovski 2017; Farahmand 2018; Schrittwieser et al. 2020; Grimm, André Barreto, et al. 2021), we propose to view both MuZero and IterVAML as members of a family of algorithms, which we call *Latent Decision Aware Model Based Actor Critic* framework (λ-AC) for convenience. A λ-AC algorithm is composed of three primary components: a latent model learning architecture, a decision aware model learning loss, and an actor-critic algorithm to obtain a control policy.[2]

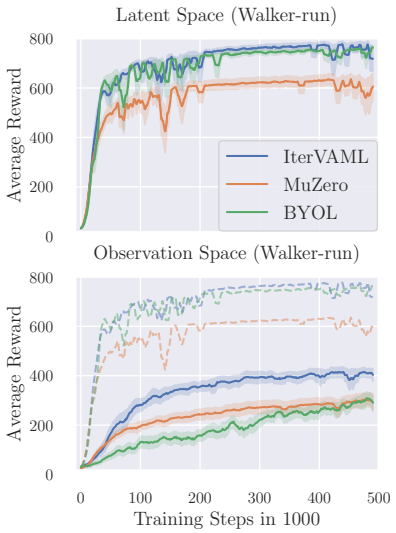

Figure 6.2: Comparison on the use of an explicit latent space with different loss functions. All losses improve in performance from the addition of an explicit latent space transformation. Dashed lines represent the mean of the latent space result. The experiments are repeated over eight random seeds and shaded areas mark three σ of the standard error of the mean over seeds.

In this work, we consider the MuZero and IterVAML model learning losses, with the MuZero value function learning scheme and model-based bootstrap. We are interested in understanding what components of the algorithms lead to performance differences in practice. To clarify the nomenclature, we designate the combination of the MuZero model and value function learning algorithm combined with the SVG objective as λ-MuZero, and the combination of IterVAML, model-based MVE and SVG as λ-IterVAML.

6.2.1 The importance of latent spaces for applied decision-aware model learning

One of the major differences between MuZero and IterVAML is that the former uses a latent model design,

[2]For an overview of the differences and similarities of different value aware loss, see Section A.4.

while the IterVAML algorithm is presented without a specific architecture. When used with a state-space prediction model, IterVAML has been shown to diverge, leading to several works claiming that it is an impractical algorithm (Lovatto et al. 2020; Voelcker, Liao, et al. 2022). However, there is no a priori reason why a latent model should not be used with IterVAML, or why MuZero has to be used with one. In prior work, sharp value function gradients have been hypothesized to be a major concern for IterVAML (Voelcker, Liao, et al. 2022). The introduction of a learned representation space in the model through a latent embedding can help to alleviate this issue.

The full loss for each experiment is $\mathcal{L}^n_{\text{DecisionAware}} + \mathcal{L}^n_{\text{latent}} + \mathcal{L}^n_{\text{reward}}$, where $\mathcal{L}_{\text{reward}}$ is an MSE term between the predicted and ground truth rewards. As a baseline, we drop the decision-aware loss and simply consider BYOL. Full pseudocode is found in **??**. As is evident from Figure 6.2 the latent space is highly beneficial to achieve good performance. For all of our experiments we use eight random seeds and the shaded areas mark three σ of the standard error of the mean over random seeds, which represents a 99.7% certainty interval under a Gaussian assumption.

This highlights that negative results on the efficacy of IterVAML are largely due to suboptimal choices in the model implementation, not due to limitations of the algorithm. In addition, the stabilizing loss is essential for IterVAML, while it can be dropped in MuZero (Ye et al. 2021) (compare **??**). We conjecture that MuZero is more stable without auxiliary losses due to incorporating the true reward into its joint model-value function loss, an advantage that IterVAML does not have. The weak performance of MuZero on walker-run sepcifically seems to be an outlier, compare Section 6.4.

6.3 Analysis of decision-aware losses in stochastic environments

While the two losses behave similarly in deterministic environments, it is an open question if this holds in stochastic ones. While previous work has noted deterministic models to be insufficient in stochastic cases (Antonoglou et al. 2022), we presented a theoretical treatment and an analysis of the loss functions of both MuZero and IterVAML. Proofs are found in Subsection A.3.1.

In Proposition 8 we establish that λ-IterVAML leads to an unbiased solution in the infinite sample limit, even when restricting the model class to deterministic functions under measure-theoretic conditions on the function class. This is a unique advantage of value-aware models. Algorithms such as Dreamer (**Hafner2020Dream**) or MBPO (Janner et al. 2019a) require stochastic models.

We then show in Proposition 9 that MuZero's joint model- and value function learning algorithm leads to a biased solution, even when choosing a probabilistic model class that contains the ground-truth environment. This highlights that while the losses are similar in deterministic environments, the same does not hold in the stochastic case. Finally, we verify the theoretical results empirically by presenting stochastic extensions of environments from the DMC suite (Tunyasuvunakool et al. 2020b).

6.3.1 Restricting IterVAML to deterministic models

In most cases, deterministic function approximations cannot capture the transition distribution on stochastic environments. However, it is an open question whether a deterministic model is sufficient for learning a *value-equivalent* model, as conjectured by (Oh, Singh, and H. Lee 2017). We answer this now in the affirmative. Showing the existence of such a model relies on the continuity of the transition kernel and involved functions ϕ and V.[3]

Proposition 8. *Let $\mathcal{X}$ be a compact, connected, metrizable space. Let p be a continuous kernel from $\mathcal{X}$ to probability measures over $\mathcal{X}$. Let $\mathcal{Z}$ be a metrizable space. Consider a bijective latent mapping $\phi : \mathcal{X} \to \mathcal{Z}$ and any $V : \mathcal{Z} \to \mathbb{R}$. Assume that they are both continuous. Denote $V_{\mathcal{X}} = V \circ \phi$.*

Then there exists a measurable function $f^* : \mathcal{Z} \to \mathcal{Z}$ such that we have $V(f^*(\phi(x))) = \mathbb{E}_p\left[V_{\mathcal{X}}(x')|x\right]$ for all $x \in \mathcal{X}$.

We can conclude that given a sufficiently flexible function class $\mathcal{F}$, IterVAML can recover an optimal deterministic model for value function prediction. Note that our conditions solely ensure the *existence* of a measurable function; the learning problem might still be very challenging. Nonetheless, even without the assumption that the perfect model f is learnable, the IterVAML loss finds the function that is closest to it in the mean squared error sense, as shown by Farahmand (2018).

6.3.2 Sample bias of MuZero' value function learning algorithm

MuZero is a sound algorithm for deterministic environments, however, the same is not true for stochastic ones. While changed model architecture for MuZero have been proposed for stochastic cases (Antonoglou et al. 2022), the value function learning component contains its own bias. To highlight that the problem is neither due to the n-step formulation nor due to suboptimal architecture or value function class, we show that the value function estimate does not converge to the model Bellman target using the MuZero loss. Intuitively, the problem results from the fact that two samples drawn from the model and from the environment do not coincide, even when the true model and the learned model are equal (see Figure 6.3). This bias is similar to the double sampling issue in Bellman residual

[3] To reduce notational complexity, we ignore the action dependence in all following propositions; all results hold without loss of generality for the action-conditioned case as well.

minimization, but is distinct as the introduction of the stop-gradient in MuZero's loss function does not address the problem.

Proposition 9. *Assume a non-deterministic MDP with a fixed, but arbitrary policy π, and let p be the transition kernel. Let $\mathcal{V}$ be an open set of functions, and assume that it is Bellman complete: $\forall V \in \mathcal{V} : \mathcal{T}V \in \mathcal{V}$.*

Then for any $V' \in \mathcal{V}$ that is not a constant function, $\mathcal{T}V' \notin \arg\min_{\hat{V}\in\mathcal{V}} \mathbb{E}_{\mathcal{D}}\left[\hat{\mathcal{L}}^1_{MuZero}(p, \hat{V}; \mathcal{D}, V')\right]$.

The bias indicates that MuZero will not recover the correct value function in environments with stochastic transitions, even when the correct model is used and the function class is Bellman complete. On the other hand, model-based MVE such as used in λ-IterVAML can recover the model's value function in stochastic environments.

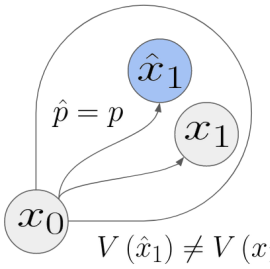

Figure 6.3: The source of the MuZero bias. Though x_1 and $\hat{x}_1$ are drawn from the same distribution, their values do not coincide.

The bias is dependent on the variance of the value function with regard to the transition distributions. This means that in some stochastic environments the MuZero loss might still perform well. But as the variance of the value function increases, the bias to impact the solution.

If the MuZero loss is solely used for model learning and the value function is learned fully model-free or model-based, the IterVAML and MuZero algorithms show strong similarities (compare Subsection A.3.2). The main difference is that MuZero uses a bootstrap estimate for the value function, while IterVAML uses the value function estimate directly. However, when jointly training value function and model in a stochastic environment, neither the model nor the value function converge to the correct solution, due to the tied updates.

6.3.3 Empirical validation of the performance in stochastic environments

To showcase that the bias of MuZero's value learning strategy is not merely a mathematical curiosity, we tested the two losses in the challenging humanoid-run tasks from the DMC benchmarking suite (Tunyasuvunakool et al. 2020b) with different noise distributions applied to the action (details on the environment can be found in **??**).

As shown in Figure 6.4, we do see a clear difference between the performance of λ-IterVAML and λ-MuZero when increasing the noise level. At small levels of noise, all algorithms retain their performance. This is expected, since small noise on the actions can increase the robustness of a learned policy or improve exploration (Hollenstein et al. 2022). At large levels of noise, however, the λ-MuZero algorithm drops in performance to the value-unaware

baseline. We provide additional experiments with more stochastic environments in **??** which further substantiate our claims.

It is important to note that humanoid-run is a challenging environment and strong noise corruption increases difficulty, therefore λ-IterVAML will not retain the full performance in the stochastic version. Furthermore, as highlighted before, the stabilizing latent prediction loss is necessary for λ-IterVAML and introduces some level of bias (for details see Subsection A.3.4). However, as all algorithms are impacted by the stabilization term, it is still noticeable that λ-MuZero's performance drops more sharply. This raises an important direction for future work, establishing a mechanism for stabilizing value-aware losses that does not introduce bias in stochastic environments.

6.4 Evaluating model capacity and environment choice

After investigating the impact of action noise on different loss function, we now present empirical experiments to further investigate how different implementations of λ-AC behave in empirical settings. Theoretical results (Farahmand, André Barreto, and Nikovski 2017; Farahmand 2018) show that value-aware losses perform best when learning a correct model is impossible due to access to finite samples or model capacity. Even though the expressivity of neural networks has increased in recent years, we argue that such scenarios are still highly relevant in practice. Establishing the necessary size of a model a priori is often impossible, since RL is deployed in scenarios where the complexity of a solution is unclear. Increasing model capacity often comes at greatly increased cost (Kaplan et al. 2020), which makes more efficient alternatives desirable. Therefore, we empirically verify under what conditions decision-aware losses show performance improvements over the simple BYOL loss.

To replace the MCTS based policy in discrete state-space MuZero, we introduced the use of model value gradients with decision-aware losses. However, all loss functions were originally designed only for improving value function estimation. We verify that they still provide

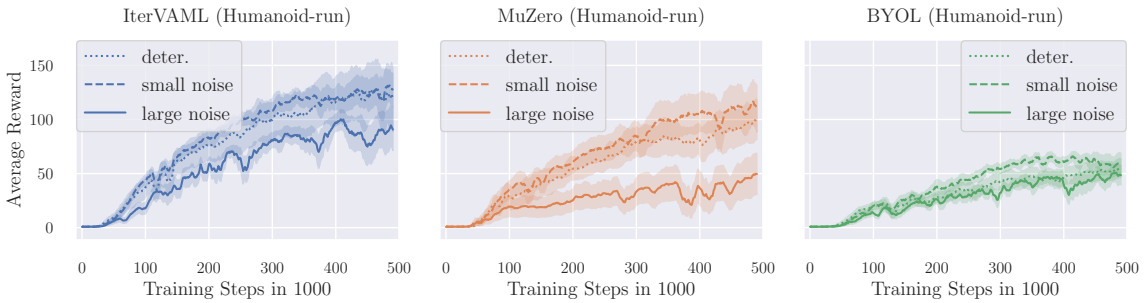

Figure 6.4: Comparison of the impact of noise across all algorithms. While both IterVAML and MuZero are impacted by the level of noise, we observe a larger drop in MuZero, which does not perform above the BYOL baseline at the highest noise level.

benefit for policy gradient approximation by replacing the model-based policy gradient with a model-free version.

For these experiments, we use environments from the popular DMC suite (Tunyasuvunakool et al. 2020b), Walker-run, Quadruped-run, and Humanoid-run. These environments were picked from three different levels of difficulty (N. Hansen, X. Wang, and Su 2022) and represent different challenge levels in the benchmark, with Humanoid-run being a serious challenge for most established algorithms.

The algorithmic framework and all model design choices are summarized in **??**. The model implementation follows N. Hansen, X. Wang, and Su (2022). Reported rewards are collected during training, not in an evaluation step, the same protocol used by N. Hansen, X. Wang, and Su (2022).

When do decision-aware losses show performance improvements over the simple BYOL loss? The results in Figure 6.5 show that both MuZero and IterVAML provide a benefit over the BYOL loss in the most challenging environment, Humanoid-run. This is expected, as modelling the full state space of Humanoid-run is difficult with the model architectures we used. For smaller networks, we see a stable performance benefit in Figure 6.6 from the MuZero loss. λ-IterVAML also outperforms the BYOL baseline, but fails to achieve stable returns.

This highlights that in common benchmark environments and with established architectures, value-aware losses are useful in challenging, high-dimensional environments. However, it is important to note that we do not find that the IterVAML or MuZero losses are harmful in deterministic environments, meaning a value-aware loss is always preferable.

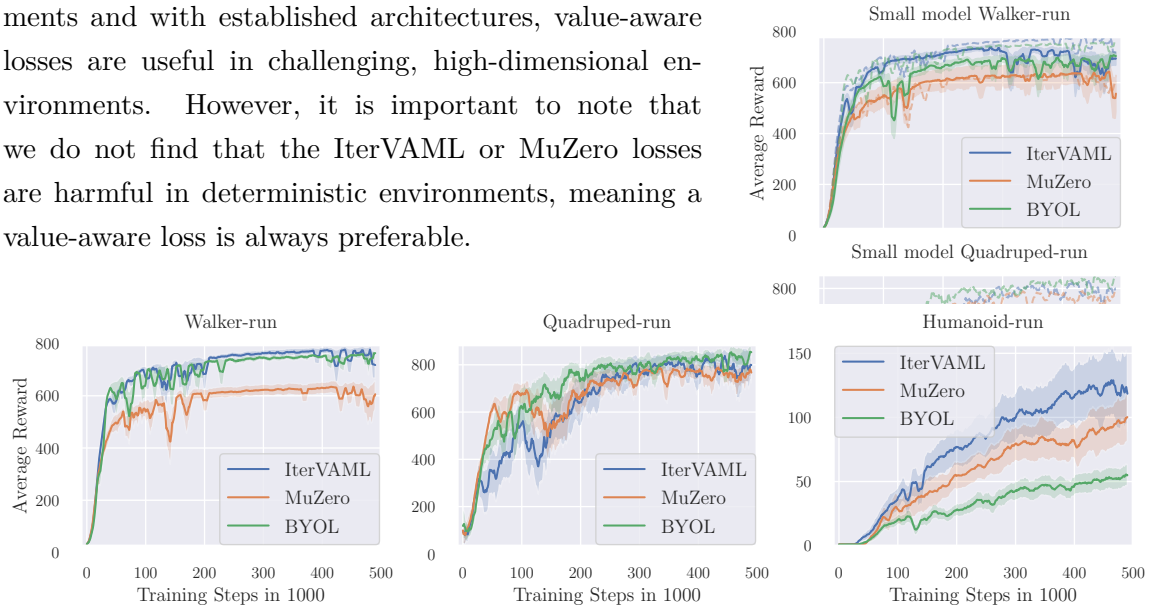

Figure 6.5: Performance comparison overall test environments. We see that IterVAML performs slightly above MuZero in several environments, but decisive gains from the value-aware losses (λ-IterVAML, λ-MuZero) over BYOL can only be observed in the challenging Humanoid-run environment.

more impacted by decreasing model size on more challenging tasks, due to the lack of real reward signal in the value function loss. Humanoid is omitted as no loss outperforms a random baseline. Dashed lines show the mean results of the bigger models for comparison.

The performance improvement of MuZero over IterVAML with very small models is likely due to the effect of using real rewards for the value function target estimation. In cases where the reward prediction has errors, this can lead to better performance over purely model-based value functions such as those used by IterVAML.

Can decision-aware models be used for both value function learning and policy improvement? In all previous experiments, the models were used for both value function target estimation and for policy learning with a model-based gradient. To investigate the performance gain from using the model for policy gradient estimation, we present an ablation on all environments by substituting the model gradient with a simple deep deterministic policy gradient.

As seen in Figure 6.7, all losses and environments benefit from better policy estimation using the model. Therefore it is advisable to use a λ-AC model both for gradient estimation and value function improvement. Compared to MuZero, IterVAML loses more performance without policy gradient computation in the hardest evaluation task. It has been noted that model-based value function estimates might be more useful for policy gradient estimation than for value function learning (Amos et al. 2021; Ghugare et al. 2023). In addition, the grounding of the MuZero loss in real rewards leads to better value function prediction in the Humanoid-run environment. Therefore a better gradient can be obtained with MuZero when the model is not used for policy gradient estimation, since a correct value function estimate becomes more important.

6.5 Related work

VAML and MuZero Farahmand (2018) established the IterVAML formulation based on earlier work (Farahmand, André Barreto, and Nikovski 2017). Several extensions based on this formulation have been proposed, such as a VAML-regularized MSE loss (Voelcker, Liao, et al. 2022) and a policy-gradient aware loss (Abachi, Ghavamzadeh, and Farahmand 2020). Combining IterVAML with latent spaces was first developed by Abachi, Voelcker, et al. (2022), but no experimental results were provided. MuZero (Schrittwieser et al. 2020; Ye et al. 2021) is build based on several earlier works which introduce the ideas of learning a latent model jointly with the value function (Silver, H. v. Hasselt, et al. 2017;

Oh, Singh, and H. Lee 2017). However, to the best of our knowledge, none of these works highlight the importance of considering the bias in stochastic environments that result from such a formulation. Antonoglou et al. (2022) propose an extension to MuZero in stochastic environments, but focus on the planning procedure, not the biased value function loss. N. Hansen, X. Wang, and Su (2022) adapted the MuZero loss to continuous control environments but did not extend their formulation to stochastic variants. Grimm, André Barreto, et al. (2020) and Grimm, André Barreto, et al. (2021) consider how the set of value equivalent models relates to value functions. They are the first to highlight the close connection between the notions of value-awareness and MuZero.

Other decision-aware algorithms Several other works propose decision-aware variants that do not minimize a value function difference. D'Oro, Metelli, et al. (2020) weigh the samples used for model learning by their impact on the policy gradient. Nikishin, Abachi, et al. (2022) uses implicit differentiation to obtain a loss for the model function with regard to the policy performance measure. To achieve the same goal, Eysenbach et al. (2022) and Ghugare et al. (2023) choose a variational formulation in the control-as-inference framework. **Modhe2021ModelAdvantageOF** proposes to compute the advantage function resulting from different models instead of using the value function. Ayoub et al. (2020) presents an algorithm based on selecting models based on their ability to predict value function estimates and provide regret bounds with this algorithm.

Learning with suboptimal models Several works have focused on the broader goal of using models with errors without addressing the loss functions of the model. Among these, several focus on correcting models using information obtained during exploration (Joseph et al. 2013; E. Talvitie 2017; Modi et al. 2020; Rakhsha et al. 2022), or limiting interaction with wrong models (Buckman et al. 2018; Janner et al. 2019a; Abbas, Sokota, et al. 2020). Several of these techniques can be applied to improve the value function estimate of a λ-AC world model further. Finally, we do not focus on exploration in this chapter, but Z. D. Guo et al. (2022) show that a similar loss to ours can be exploited for targeted exploration.

6.6 Conclusions

In this chapter, we investigated the λ-AC framework for model-based reinforcement learning with decision-aware models. The core components of all algorithms in this framework are a) a latent world model, b) a value-aware loss function and c) a model-based actor-critic approach. Empirically, we show that the design decisions established for MuZero are useful for all λ-AC algorithms. Previously established performance differences (Lovatto et al. 2020; Voelcker, Liao, et al. 2022) can be overcome with latent model architectures. We furthermore establish a formal limitation on the performance of MuZero in stochastic

environments, and verify this empirically. Finally, we conduct a series of experiments to establish which algorithmic choices lead to good performance in empirical settings.

Our results highlight the importance of decision-aware model learning in continuous control and allow us to make algorithmic recommendations. When the necessary capacity for an environment model cannot be established, using a decision-aware loss will improve the robustness of the learning algorithm with regard to the model capacity. In deterministic environments with deterministic models, MuZero's value learning approach can be a good choice, as the use of real rewards provides a grounded learning signal for the value function. In stochastic environments, a model-based bootstrap is more effective, as the model-based loss does not suffer from MuZero's bias.

Overall, we find that λ-AC is an important addition to the RL toolbox in complex environments where other modelling approaches fail. In future work, evaluating other design decisions, such as probabilistic models (Ghugare et al. 2023) and alternative exploration strategies (N. Hansen, X. Wang, and Su 2022; Z. D. Guo et al. 2022) can provide important insights.

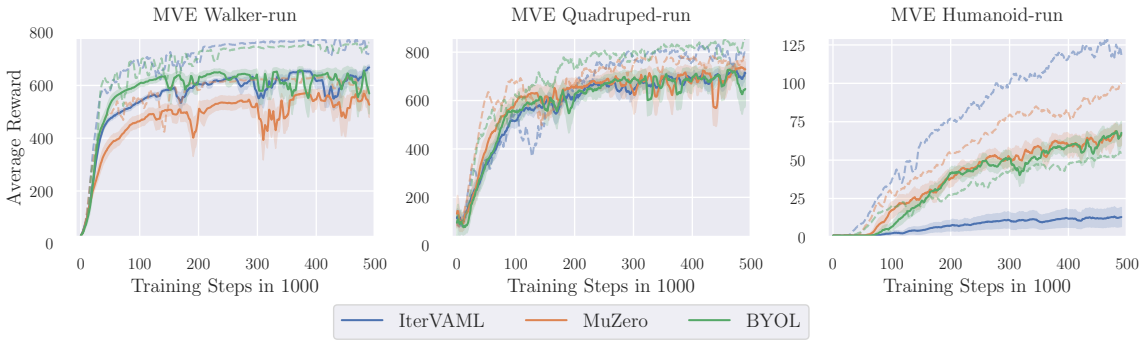

Figure 6.7: Comparison of the impact removing model policy gradients in all environments. Overall we see decreases in performance across all environments and loss functions, with a drastic decrease in performance for IterVAML in the hardest task. Dashed lines show the mean results of the model-based gradient for comparison.

Chapter 7

Stabilizing Value Function Learning with Model-Generated Data

This chapter is based on Claas A Voelcker, Marcel Hussing, Eric Eaton, Amirmassoud Farahmand, and Igor Gilitschenski (2025). "MAD-TD: Model-Augmented Data stabilizes High Update Ratio RL". in: *International Conference on Learning Representations*.

7.1 Introduction

Instead of solely relying on data gathered by a target policy, *off-policy* reinforcement learning (RL) aims to leverage experience gathered by past policies (Richard S. Sutton and Barto 2018b) to fit a value function for the target policy. Ideally, training over many iterations should help extract important information from past data. However, recent work has shown that simply increasing the number of gradient update steps, the *replay ratio* or *update-to-data (UTD) ratio*, can lead to highly unstable learning (Nikishin, Schwarzer, et al. 2022; D'Oro, Schwarzer, et al. 2023; Hussing et al. 2024; Nauman, Ostaszewski, et al. 2024).

Prior work has stabilized the learning by using double Q minimization to reduce overestimation (Fujimoto, Hoof, and Meger 2018), training ensemble methods to improve value estimation (Xinyue Chen, C. Wang, Zhou, and Keith W Ross 2020; Hiraoka et al. 2022), introducing architectural regularization (Hussing et al. 2024; Nauman, Ostaszewski, et al. 2024), or resetting networks periodically throughout the learning process (D'Oro, Schwarzer, et al. 2023; Schwarzer, Obando Ceron, et al. 2023; Nauman, Ostaszewski, et al. 2024). However, while useful, pessimistic underestimation and architectural regularization are in-

sufficient by themselves to combat the problem (Hussing et al. 2024), and so most methods resort to either network resets or ensembles. Critic ensembles can be expensive to train, and resetting has several important limitations: in real systems, re-executing a random policy can be expensive or unsafe, the resetting interval needs to be carefully tuned (Hussing et al. 2024), and when storing a full reset buffer is infeasible, resetting loses important information.

We narrow in on a key issue contributing to unstable training: *wrong value function estimation on unobserved on-policy actions* (Thrun and Schwartz 1993; Tsitsiklis and Van Roy 1996). Off-policy RL uses the values of states sampled under old policies with actions from the target policy to update the value function. However, these state-action pairs themselves are not in the replay buffer and hence their value estimate is not directly improved by training. Consequently, a learned function which achieves low error on seen data is not guaranteed to generalize well to actions that *differ* from past actions. This problem is related to overfitting (Li et al. 2023) and contributes to overestimation (Thrun and Schwartz 1993; H. v. Hasselt 2010; Fujimoto, Hoof, and Meger 2018). However, overfitting assumes that train and test set are drawn from the same distribution, while we focus on the distribution shift between data collection and target policy. Previous work has investigated the difficulty of off-policy learning due to this shift (Maei et al. 2009; Richard S Sutton, Mahmood, and White 2016; H. v. Hasselt 2010; Fujimoto, Hoof, and Meger 2018), yet there are no tractable mitigation strategies that work well in the high UTD regime with deep RL.

To corroborate our hypothesis that generalization to unobserved actions is a major obstacle for training at high UTDs, we examine the behavior of value functions on on-policy transitions. Our experiments reveal that value functions generalize significantly worse to unobserved on-policy action transitions than to validation data from the same distribution as the training set. Building on this, we propose to improve on-policy value estimation by using *model-generated on-policy data*.

Previous investigations into model-based deep RL have focused on learning values fully in model roll-outs (**Hafner2020Dream**; Buckman et al. 2018; Janner et al. 2019b; Ghugare et al. 2023) and the associated challenges (Y. Zhao et al. 2023; N. Hansen, Su, and X. Wang 2024). In contrast, we show that mixing a small amount of model-generated on-policy data with real off-policy replay data is sufficient to achieve stable learning in the high UTD regime. Our method, Model-Augmented Data for TD learning (MAD-TD), mitigates the generalization issues of the value function in the hardest tasks of the DeepMind control (DMC) benchmark (Tunyasuvunakool et al. 2020a) and achieves strong and stable high UTD learning without resetting or redundant ensemble learning.

The main contributions of this work are twofold: First, we empirically show the existence of

misgeneralization from off-policy value estimation to on-policy predictions. We connect this issue to the challenge of stable learning with high update ratios and highlight how increasing the update ratio increases Q function overestimation. Second, we provide a new method called MAD-TD that improves the value function accuracy on unobserved on-policy actions with model-generated data and stabilizes training at high update ratios. This method proves to have equivalent performance to or outperform previous strong baselines.

7.2 Mathematical background

We consider a standard RL setting, the discounted infinite-horizon MDP $(\mathcal{X}, \mathcal{A}, \mathcal{P}, r, \rho, \gamma)$ with state space $\mathcal{X}$, action space $\mathcal{A}$, a transition kernel $\mathcal{P} : \mathcal{X} \times \mathcal{A} \to \mathcal{M}(\mathcal{X})$, a reward function $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, starting state distribution $\rho \in \mathcal{M}(\mathcal{X})$ and a discount factor $\gamma \in [0, 1)$ (Puterman 1994; Richard S. Sutton and Barto 2018b). For a space Y we use $\mathcal{M}(Y)$ to denote the set of probability measures over the space. Our goal is to learn a policy $\pi : \mathcal{X} \to \mathcal{M}(\mathcal{A})$ that maximizes the discounted sum of future rewards

$$\pi^* \in \arg\max_{\pi \in \Pi} \sum_{t=0}^{\infty} \mathbb{E}_{\mathcal{P}^\pi}[\gamma^t r(x_t, a_t) | x_0 \sim \rho] \ ,$$

where actions are sampled according to the policy and new states according to the transition kernel.

7.2.1 Off-policy value function learning

As an intermediate objective, many algorithms attempt to simplify the direct policy optimization problem by first learning a policy value function Q^π, which is defined via a recursive equation

$$Q^\pi(x, a) = r(x, a) + \gamma \mathbb{E}_{x' \sim \mathcal{P}(\cdot|x,a), a' \sim \pi(\cdot|x')} \left[Q^\pi(x', a') \right] \ .$$

The policy can then be incrementally improved by picking $\pi_{k+1}(x) \in \arg\max_{a \in \mathcal{A}} Q^{\pi_k}(x, a)$ at every time step k. In practice, Q^π and π are often parameterized as neural networks and learned from data. To increase the sample efficiency of the algorithm, it is common to store *all* collected interaction data independent of the collection policy in a replay buffer $\mathcal{D} = \{(x_t, a_t, r_t, x_{t+1})_{t=0}^T\}$. As the Q-value only depends on the policy via the policy evaluation at the next state, it is possible to estimate Q-values from past interaction data by minimizing the fitted Q-learning objective

$$\mathcal{L}\left(\hat{Q} \middle| \mathcal{D}, \pi\right) = \frac{1}{|\mathcal{D}|} \sum_{t=0}^{T} \left| \hat{Q}(x_t, a_t) - \left[r_t + \gamma \hat{Q}\left(x_{t+1}, a'\right) \right]_{\text{sg}} \right|^2 \quad \text{with } a' \sim \pi(\cdot|x_{t+1}) \ .$$

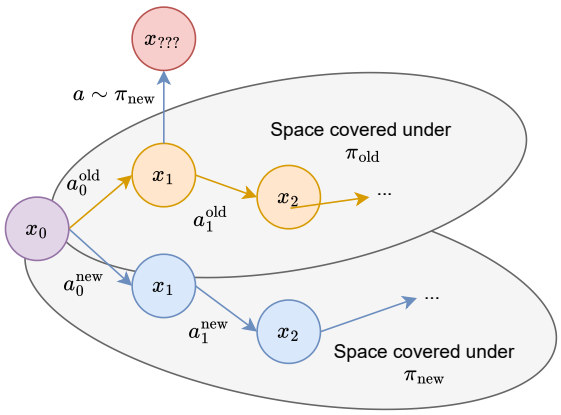

Figure 7.1: A visualization of the core issue we investigate. Even if a replay buffer contains good coverage for two policies (π_{old} and π_{new}) starting from $\rho = x_0$, this does not guarantee that it contains a transition for executing an action under the new policy on a state visited under the old. However, this state-action pair's value estimate is used to update the value of state x_0 via Subsection 7.2.1, without being grounded in an observed transition.

Here $[\cdot]_{\text{sg}}$ denotes the stop gradient operation introduced to avoid the double sampling bias and all data contained in the replay buffer is colored blue. However, the Q value at the next state x_{t+1} is evaluated with an action a' that is *not* guaranteed to be in the replay memory, as the target policy can be different from the policy used to gather the sample. This means that we require the Q value to generalize to potentially unseen actions. We provide a visualization of this issue in Figure 7.1.

7.3 Investigating the root cause of unstable Q learning

Minimizing Subsection 7.2.1 finds the policy Q function over a replay buffer with sufficient coverage of all states and actions that this policy visits. However, in most continuous control RL algorithms (Lillicrap et al. 2016; Haarnoja et al. 2018a; Fujimoto, Hoof, and Meger 2018), this update is interleaved with policy update steps . The data in $\mathcal{D}$ then necessarily becomes *off-policy* as training progresses.

This means that the number of actor and critic optimization steps needs to be balanced with gathering new data. Obtaining new on-policy data is vital to continually improve policy performance (Ostrovski, Castro, and Dabney 2021), but performing more update steps before gathering new data ensures that the existing data has been used effectively to improve the policy. The *replay ratio* (Fedus et al. 2020) or *update-to-data (UTD) ratio* (Nikishin, Schwarzer, et al. 2022), which governs the number of gradient steps per environment step, is therefore a vital hyperparameter.

Naively training with high UTD ratios can lead to collapse in off-policy deep RL (Nikishin, Schwarzer, et al. 2022). We conjecture that one of the major causes of the instability of high UTD off-policy learning are wrong Q values on *unobserved actions*. This is a well-known problem for off-policy TD learning (Baird 1995; Tsitsiklis and Van Roy 1996; Richard S

Sutton, Mahmood, and White 2016; Ghosh and Marc G Bellemare 2020). To differentiate the problem from *overfitting* to the training distribution, we use the term *misgeneralization* to highlight the importance of the distribution shift in causing the issue. Our experiments in Subsection 7.3.2 show that generalization to on-policy actions is more difficult than generalization to a validation dataset that follows the training distribution, and that higher UTDs exacerbate the issue.

7.3.1 Action distribution shift can cause off-policy Q value divergence

To highlight the role that on-policy actions play in stabilizing Q value learning, we show an analysis the stability of Q learning with linear features. The core ideas follow Richard S Sutton, Mahmood, and White (2016) and are also explored by Tsitsiklis and Van Roy (1996) and Richard S Sutton (1988). We assume that the Q function is parameterized with fixed features and weights as $Q(x, a) = \phi(x, a)^\top \theta$. Let X and A be the sizes of the state and action space respectively. Let $P \in \mathbb{R}^{X \cdot A \times X}$ be the matrix of transition probabilities from state-action pairs to states. A policy can then be expressed as a mapping $\Pi \in \mathbb{R}^{X \times X \cdot A}$ from states to the likelihood of taking each action. $R \in \mathbb{R}^{X \cdot A}$ is the vector of rewards. $D^\pi \in \mathbb{R}^{X \cdot A \times X \cdot A}$ is a matrix where the main diagonal contains the discounted state-action occupancies of P^π starting from ρ. If we assume access to a mixed replay buffer $\mathcal{D} = \bigcup\{D^{\pi_1}, \ldots, D^{\pi_n}\}$ gathered with different policies, the Q learning loss for a target policy Π can be written as

$$L(\theta) = \sum_{i=1}^{n} \left[D^{\pi_i} \left(\Phi^\top \theta - [R + \gamma P \Pi \Phi^\top \theta]_{\text{sg}} \right))^2 \right] \ .$$

The stability of learning with this loss can be analyzed using the gradient flow

$$\dot{\theta} = -2\Phi \sum_{i=1}^{n} D^{\pi_i} \left(I - \gamma P \Pi \right) \Phi^\top \theta + 2\Phi \sum_{i=1}^{n} D^{\pi_i} R \ .$$

This gradient flow is guaranteed to to be stable around a fixed point θ^* if the key matrix $\sum_{i=1}^{n} D^{\pi_i} \left(I - \gamma P \Pi \right)$ is positive definite (Richard S Sutton 1988). Details and a proof of the following statement are provided in **??**. We can decompose the key matrix and see that the positive definiteness depends on the difference in policy between the replay buffer and the target policy

$$\sum_{i=1}^{n} D^{\pi_i} \left(I - \gamma P \Pi \right) = \underbrace{\sum_{i=1}^{n} D^{\pi_i} \left(I - \gamma P \Pi_i \right)}_{\text{positive definite}} + \gamma \underbrace{\sum_{i=1} D^{\pi_i} P (\Pi_i - \Pi)}_{\text{no guarantees}} \ .$$

In general, we can provide no guarantees for the second term outside of the on-policy case ($\Pi_i = \Pi$) where it becomes 0. The stability depends on the difference between the target

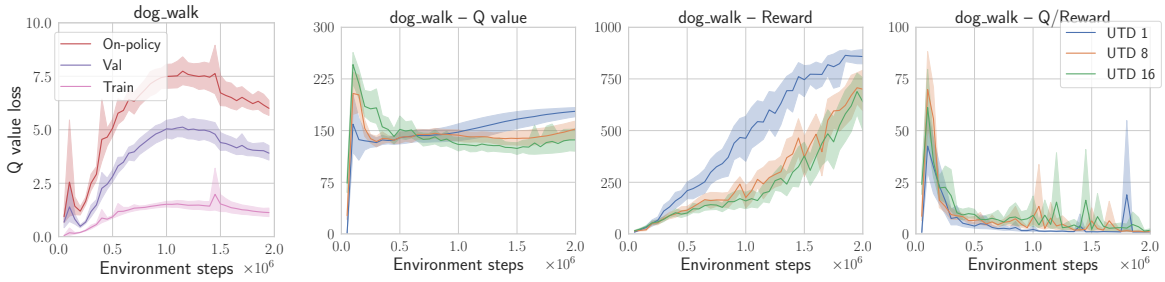

Figure 7.2: Left: the train, validation, and on-policy validation error of the Q function at UTD 1. Right: the Q values and return curves of TD3 agents across different UTD 1, 8, and 16.

policy and the data-collection policies. If the target policy takes actions which are not well covered under the training policies, the remainder can be non positive definite. This also matches the intuition that learning fails if we simply do not have sufficient evidence for the Q function of unobserved actions.

When using features, the eigenvalue conditions on the key matrix are only sufficient, not necessary, as the features can allow for sufficient generalization between observed and unobserved state-action pairs. In deep RL, the features ϕ are updated alongside with the weights, making it hard to provide definitive mathematical statements on stability. With good function approximation, we could hope that the learned value function generalizes correctly to unseen actions. In the next section we investigate this for a non-trivial task from the DMC suite and highlight that, while the value function does not diverge irrecoverably, good generalization is not guaranteed either.

7.3.2 Empirical Q value estimation with off-policy data

In environments with large state-action space, ensuring coverage is difficult. To investigate whether learning is stable nonetheless, we train a model-free TD3 agent on the *dog walk* environment (Tunyasuvunakool et al. 2020b). The architecture is presented in Subsection 7.4.1, and is regularized to prevent catastrophic divergence (Hussing et al. 2024; Nauman, Bortkiewicz, et al. 2024) and uses clipped double Q learning (Fujimoto, Hoof, and Meger 2018). This means it uses the most common techniques which are designed to prevent misgeneralization and overestimation.

While training a TD3 agent (Fujimoto, Hoof, and Meger 2018), we save transitions in a validation buffer with a 5% probability. At regular intervals we compute the critic loss on this validation set. In addition, we reset our simulator to each validation state and sample an action from the target policy. We then simulate the ground truth on-policy transition and compute the loss over these. This allows us to test how well our value function generalizes to target policy state-action pairs (as depicted in Figure 7.1).

The results are presented in Figure 7.2 and show a gap both between the train and validation sets, as well as the validation and the on-policy sets. While we use the on-policy state-actions to update the Q value, these estimates are not actually consistent with the environment. Furthermore, the Q value overestimation grows with increasing UTDs. This phenomenon was previously discussed in the context of over-training on limited data (Hussing et al. 2024) .

The experiments show that the problem outlined in Subsection 7.3.1 is not merely a mathematical curiosity, but that Q value generalization to out-of-replay-distribution actions is difficult in practice, and becomes more difficult with increasing update ratios. Even though full divergence is not observed as new data is continually added to the replay buffer, it takes a long time for the effects of severe early overestimation to dissipate.

7.3.3 Previous attempts to combat misgeneralization and overestimation

Prior strategies that deal with misgeneralization can be grouped into three major directions: architectural regularization to prevent divergence of the value function, pessimism or ensemble learning to combat overestimation, and networks resets to restart learning. While all of these interventions help to some degree, they each either do not solve the problem in full or cause additional issues.

Architectural regularization Architecture changes (Hussing et al. 2024; Nauman, Bortkiewicz, et al. 2024; Nauman, Ostaszewski, et al. 2024; Lyle, Zheng, Khetarpal, et al. 2024) and auxiliary feature learning losses (Schwarzer, Anand, et al. 2021; Y. Zhao et al. 2023; Ni et al. 2024; Voelcker, Kastner, et al. 2024) are largely reliable interventions, and have shown to provide improvements without much drawbacks in prior work. However, as Hussing et al. (2024) and our experiment presented in Subsection 7.3.2 highlight, by themselves they can mitigate catastrophic overestimation and divergence, but do not guarantee proper generalization.

Pessimism and ensembles To combat overestimation directly, the most prominent approach in continuous action spaces is Clipped Double Q Learning (Fujimoto, Hoof, and Meger 2018). Here, a Q value estimate is obtained from two independent estimates $\hat{Q}_1$ and $\hat{Q}_2$. If the error of the two critic estimators is assumed to be independent noise on the true critic estimate then using the minimum over both estimates is guaranteed to underestimate the true critic value in expectation. However, in complex settings this assumption on the the error of the critic estimates may not hold.

Ensembles (Q. Lan et al. 2020; Xinyue Chen, C. Wang, Zhou, and Keith W Ross 2020; Hiraoka et al. 2022; Farebrother, Greaves, et al. 2023) or online tuning of the rate of pessimism (Moskovitz et al. 2021) have been proposed to obtain tighter lower bounds on the

Q value. However, these strategies can be expensive as redundant models or hyperparameter tuning are needed. As a simpler strategy, recent works have also employed clipping to obtain an upper bound of the Q function to prevent divergence (Fujimoto, Chang, et al. 2024).

Resetting Finally, network resets been shown to mitigate training problems (Nikishin, Schwarzer, et al. 2022; D'Oro, Schwarzer, et al. 2023; Schwarzer, Obando Ceron, et al. 2023; Nauman, Ostaszewski, et al. 2024) in high UTD regimes. However, in cases where the agent fails to explore any useful parts of the state space within the reset interval, restarting the learning process will not improve performance (Hussing et al. 2024). This makes tuning the resetting interval both important and potentially difficult and no tuning recipes have been presented. Resetting is also a potentially hazardous strategy in real-world applications, where re-executing a random policy might be costly or infeasible due to safety constraints. Finally, it heavily relies on the assumption that all past interaction data can be kept in the replay buffer.

All of these strategies are somewhat able to alleviate the problem of out-of-distribution value estimation, yet none of them directly address the issue at the root. In the next chapter, we present an alternate approach that aims to directly regularize the action value estimates under the target policy.

7.4 Mitigation via model-generated synthetic data

As value functions misgeneralize due to lack of sufficient on-policy data, we propose to obtain synthetic data from a learned model instead. However, model-based RL can also cause problems such as compounding world model errors and optimistic exploitation of errors in the learned model. By using both real and model-generated data, we can trade-off these issues: on-policy data improves the value function and limits the impact of off-policy distribution shifts, while using only a limited number of model-generated samples prevents model errors from deteriorating the value estimates.

Our approach builds on the TD3 algorithm (Fujimoto, Hoof, and Meger 2018) and uses an update ratio of 8 by default. Our critic is updated with both model-based and real data following the DYNA framework (Richard S Sutton 1990). More precisely, we replace a small fraction α of samples $\{x, a, r, x'\}$ in each batch with samples from a learned model $\hat{p}$ starting from the same state $\{x, \pi(x), \hat{r}, \hat{x}'\}$ with $\hat{r}, \hat{x}' \sim \hat{p}(\cdot|x, \pi(x))$. In our experiments, α is set to merely 5%. We term this approach Model-Augmented Data for TD learning (MAD-TD).

Model vs Q function generalization We expect that a learned models will yield better generalization than the Q function for two reasons. First, the policy is updated each step to

find an action that maximizes the value function. This means we are effectively conducting an adversarial search for overestimated values. The model's reward and state estimation error on the other hand are independent of this process. We test the adversarial robustness of our model-augmented value functions in Subsection 7.5.3. Second, our experiment shows that value functions primarily diverge at the beginning of training. In these cases, coverage is low and on-policy state-action pairs are often not available. Obtaining a slightly wrong, yet converging value estimate can then be more useful than a diverging one. Even as more data is gathered, new policies might not revisit old states with a high likelihood. Therefore even as training continues we expect the model data to provide some benefit.

7.4.1 Design choices and training setup

Our model is based on the successful TD-MPC2 model (N. Hansen, Su, and X. Wang 2024) combined with the deterministic actor-critic algorithm TD3 (Fujimoto, Hoof, and Meger 2018). We aim to reduce the complexity of TD-MPC2 to the minimal necessary components to achieve strong learning in the DM Control suite, and thus forgo added exploration noise, SAC, ensembled critics, and longer model rollout for training or policy search. We outline several design choices here and refer to **??** for more detail.

Encoder: Like TD-MPC2, we parameterize the state with a learned encoder $\phi : \mathcal{X} \to \mathcal{Z}$ with a SimNorm nonlinearity (Lavoie et al. 2023). This transformation groups a latent vector into groups of k entries and applies a softmax transformation over each group. This bounds the norm of the features, which has been shown aid with stable training (Hussing et al. 2024; Nauman, Bortkiewicz, et al. 2024).

Critic representation and loss: We use the HL-Gauss transformation to represent the Q function (Farebrother, Orbay, et al. 2024). The critic loss is the cross-entropy between the estimated Q function's categorical representation and the bootstrapped TD estimate. To stabilize learning, we initialize the critic network towards predicting 0 for all states. We find that this stabilizes the initial update steps in which almost no data is available.

Model loss: The world model predicts the next state latent representation and the observed reward from a given encoded state $\phi(x)$ and action a. The loss has three terms: the cross-entropy loss over the SimNorm representation of the encoded next state, the MSE between the reward predictions, and the cross-entropy between the next state critic estimate and the predicted state's critic estimate. This final term replaces the MuZero loss in TD-MPC2 with a simplified variant based on the IterVAML loss (Farahmand 2018). We provide the exact mathematical equations for the loss in **??**.

Training: We train the architecture by interleaving one environment step with one round of updates with a varying number of gradient steps governed by the UTD parameter. For each update step, a new mini-batch is sampled independently from a replay buffer of pre-

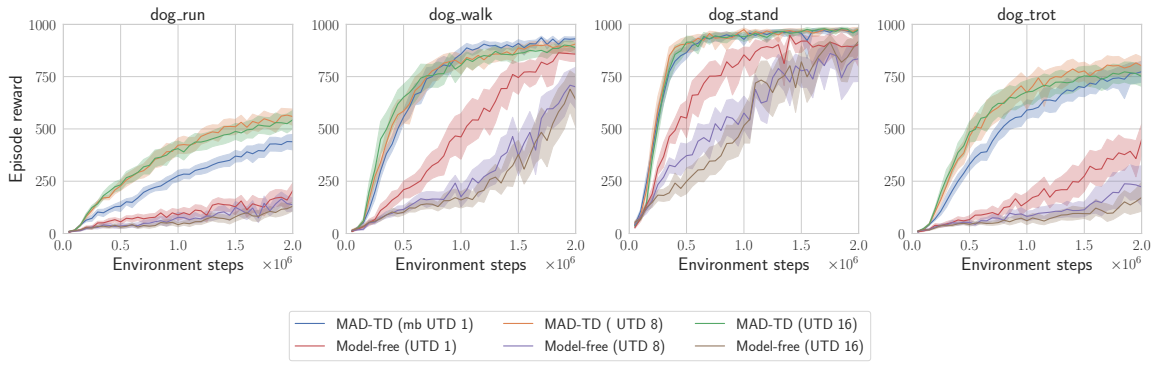

Figure 7.3: Return curves for the dog tasks with differing UTD values. Across all tested environments, the return increases or remains stable with increasing UTD when training with MAD-TD. Without model data, the performance decreases when increasing the UTD. MPC is turned off in these runs to cleanly evaluate the impact of model data on critic learning.

viously collected experience. We found that varying the number of update steps only for the critic and actor while keeping the update ratio for the model and encoder updates at 1 leads to significantly more stable learning.

Run-time policy improvement with MPC: Following the approach outlined by N. A. Hansen, Su, and X. Wang (2022), the learned model can also be used at planning time to obtain a better policy. Using the model for MPC at planning time exploits the same benefit of models as the critic learning improvement: we obtain a model-corrected estimate of the value function and choose our policy accordingly. As we only train our model for one step, we also conduct the MPC rollout for one step into the future.

7.5 Experimental evaluation

We conduct all of our experiments on the DeepMind Control suite (Tunyasuvunakool et al. 2020a). Following Nauman, Ostaszewski, et al. (2024)'s recommendations we focus our main comparisons and ablations on the two hardest settings, the *humanoid* and *dog* environments (which we will refer to as the *hard suite*). Implementation details can be found in **??**. Unless stated otherwise we evaluate MAD-TD with a UTD of 8 and use the same hyperparameters across all tasks.

Note that even though we refer to training MAD-TD without using model data for the critic as "model-free", the algorithm still benefits from the model through feature learning which has proven to be a strong regularization technique in high UTD settings (Schwarzer, Obando Ceron, et al. 2023). All main result curves are aggregated across 10 seeds per task and we plot mean and bootstrapped confidence intervals for the mean at the 95% certainty interval. For aggregated plots, we use the library provided by Agarwal et al. (2021). Results

on more environments are presented in **??**.

7.5.1 Impact of using model-generated data

We first repeat the experiment presented in Subsection 7.3.2 and show the results in Figure 7.4. Using model-based data closes the gap between on-policy and validation loss. We also observe that the initial Q overestimation disappears, which is consistent across all hard environments (see **??**). This provides evidence that we are indeed able to overcome the unseen action challenge.

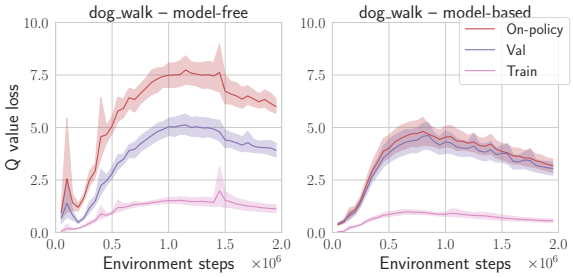

Figure 7.4: Average loss values with and without the model generated data (compare Figure 7.2) at UTD 1.

Performance with and without model data at varying UTD ratios: In Figure 7.3 we present the impact of using model-based data across different UTD ratios. Humanoid results are found in **??**. As is directly evident, across the dog tasks, we observe stagnating or deteriorating performance when increasing the update ratio, consistent with reports in prior work. However, when using a small fixed amount of model generated data, this trend is reversed across all tested environments, with performance improving or at least remaining consistent. We find that with model-based data, training is stable across a range of UTDs, even beyond those tested in recent high UTD work (Nauman, Ostaszewski, et al. 2024). We also note that we observe only limited benefits from increasing the UTD ratio when properly mitigating *misgeneralization*, except for the highly challenging dog run task, all other tasks perform on par with each other.

Comparison with baselines: As our method combines model-free and model-based updates, we compare our method against both TD-MPC2 (N. Hansen, Su, and X. Wang 2024), a strong model-based baseline, and BroNet (Nauman, Ostaszewski, et al. 2024), a recent algorithm proposed for high UTD learning. Since Nauman, Ostaszewski, et al. (2024) and N. Hansen, Su, and X. Wang (2024) trained with differing numbers of action repeats, and we found that the performance does not cleanly translate between these regimes, we present our method both with an action repeat value of 1 and 2. Some hyperparameters are adapted to the AR=1 setting (compare **??**). The results are presented in aggregate in Figure 7.5, with per environment curves show in **??**. We find that our method performs on par or above previous methods, and strikingly it is able to achieve higher returns faster than both TD-MPC2 and BRO.

7.5.2 Performance and stability impact of resetting

Resetting comparison: To investigate whether our technique benefits from more stable training, we set up a comparison in which we test the effects of resetting on our method. Figure 7.6 presents aggregate results comparing our approach and BRO, both with and without resetting. Across all tasks we find that resetting barely improves MAD-TDs performance with the tested hyperparameters and update steps. Benefits can only be observed on some seeds and can most likely be attributed to restarting the exploration process (Hussing et al. 2024). However, the BRO algorithm is not able to achieve reliable performance without resets. Overall, these results highlight that our model substantially improves the problems related to incorrect generalization of the value function, and that these are likely a major cause of the failure of high UTD learning in the DMC tasks. Conjectured problems like the primacy bias effect (Nikishin, Schwarzer, et al. 2022) need to be carefully investigated as we do not find evidence that a primacy bias impacts MAD-TD's performance in the DMC environments. Our work of course does not preclude the existence of phenomena such as loss of stability in different environments, architectures, or training setups. More discussion on this can be found in **??**.

Continued training: To highlight the pitfalls of resets, we employ a common RL theory metric the per timestep average regret

$$\overline{\text{Reg}}(T) = \frac{1}{T}\sum_{t=0}^{T-1}\left(\mathcal{R}^* - \mathcal{R}_t\right) \quad ,$$

where $\mathcal{R}_t$ denotes the cumulative return in episode t and $\mathcal{R}^*$ the optimal return. We use the maximum return any of the algorithms achieved $\hat{\mathcal{R}}^*$ as a lower bound on the optimal return $\mathcal{R}^*$. Regret quantifies how much better the algorithm could have performed throughout training. In other words, in situations where continued learning is crucial, such as many safety critical applications, regret might be a better measure of performance. It captures not only how good the final policy is, but also how well the algorithm adapts over time, and minimizes mistakes. We present a comparison of MAD-TD and the resetting-based BRO in Figure 7.7 using an action repeat of 1. The results show, even though both algorithms are close in their final return, their training behavior differs vastly. MAD-TD has lower regret showcasing its strength in continued deployment.

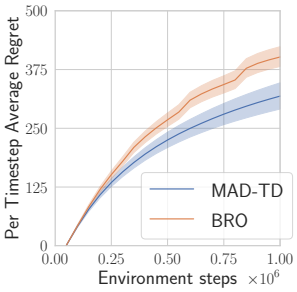

Figure 7.7: Average regret (↓), mean over the hard suite. Lower regret corresponds to faster, more stable training. MAD-TD outperforms BRO.

7.5.3 Further experiments and ablations

To further test our approach, we present two additional experiments on the *hard suite*: changing the action selection for the model data generation, and reducing the model perfor-

mance. In addition, we investigate the impact of using model based data on the smoothness of the learned value function.

Off-policy action selection in the model: To verify that the improvement in performance is due to the off-policy correction provided by the model, we repeat the *hard suite* experiments with a UTD of 8 and 5% model data, but we chose actions randomly from a uniform distribution across the action space. The results are presented in Figure 7.8. They highlight that random state-action pairs do not provide the necessary correction and the performance deteriorates to that of the model-free baseline.

Smaller model networks: To study the effect of the modelling capacity on our method, we ablate the size of the latent model by reducing the network size across the hard suite. The results are presented in Figure 7.9. We see that reducing the network size has an immediate and monotonic impact on the performance of our approach, suggesting that the model learning accuracy and prediction capacity is indeed vital for our approach to function well. However, even with small models of 64 hidden units, we still see some benefits from training with the model predicted data.

Perturbation robustness of the model-corrected value function: To motivate our method, we conjectured that one of the problems of training in actor-critic learning is that the actor is conducting a quasi adversarial search for overestimated values on the learned critic.[1] To substantiate this claim and provide additional insight into the benefits of our approach, we used the iterated projected gradient method Madry et al. 2018 to estimate the smoothness of the learned value functions across training on the humanoid environments at a UTD of 1 with and without model data.

Results are presented in Figure 7.10. Across the humanoid tasks we find that not using any model data leads to value functions with higher oscillations, either across the whole training run in humanoid_run, or in the middle of training like in stand and walk.

7.6 Conclusion

Our experiments allow us to conclude that wrong generalization of the value functions to unseen, on-policy actions is indeed a major challenge that prevents stable off-policy RL, both in theory and in practice. Model-Augmented Data for TD learning (MAD-TD) is able to leverage the learning abilities of latent self-prediction models to provide small, yet crucial amounts of on-policy transitions which help stabilize learning across the hardest DeepMind Control suite tasks. With a relatively simple model architecture and learning algorithm, this method proves to be on par with, or even outperform other strong approaches, and does

[1] *Quasi* because the actor is not constrained to find an action close to the replay buffer sample.

not rely on mechanisms such as value function ensembles or resetting which were previously conjectured to be necessary for stable learning in high UTD regimes.

Our work opens up exciting avenues for future work. The issue of poor generalization in off-policy learning can likely be tackled with other approaches such as diffusion models (Lu et al. 2024) or better pretrained foundation models, and our presented experiments provide an important baseline for such work. Furthermore, while we have purposefully kept our approach as simple as possible to validate our hypothesis, many ideas from the model-based RL community such as uncertainty quantification (Chua et al. 2018; E. J. Talvitie et al. 2024), multi-step corrections (**Hafner2020Dream**; Buckman et al. 2018), or policy gradient estimation (Amos et al. 2021) can be combined with our approach. Our insight that surprisingly little data is necessary to achieve strong correction can likely be leveraged in these other approaches as well to trade-off model errors and value function errors more carefully. Finally, while we chose the data to roll out in our models at random, our insights can likely be combined with ideas from the area of DYNA search control (**Pan2020Frequencybased**; Y. Pan et al. 2019) to select datapoints on which the correction has the most impact.

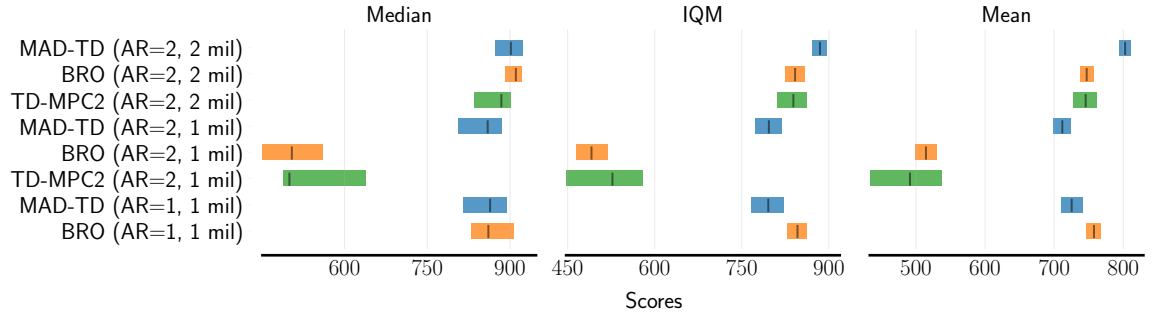

Figure 7.5: Performance comparison on the hard tasks for MAD-TD, BRO, and TD-MPC, with varying number of steps and action repeat settings. MAD-TD is on par with all baselines, has higher mean and IQM when trained for 2 million time steps and action repeat 2, and strongly outperforms TD-MPC2 and BRO at 1 million time steps with action repeat 2.

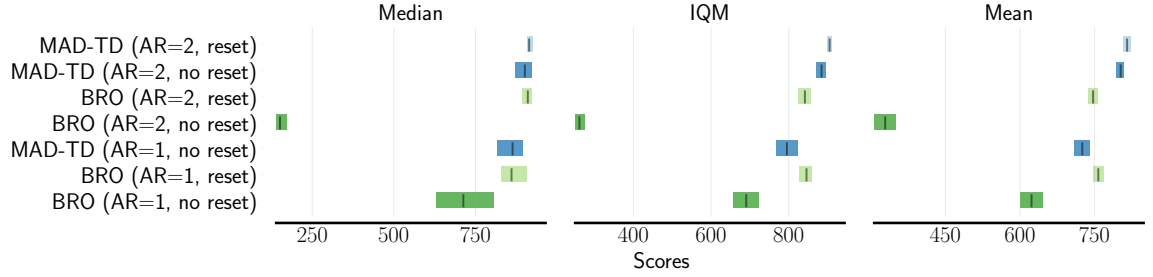

Figure 7.6: Resetting evaluation of MAD-TD and BRO. Lighter color denotes performance with reset, and darker without. While MAD-TD's performance only increases slightly when adding resetting, BRO is unable to achieve strong performance in and setting without resetting.

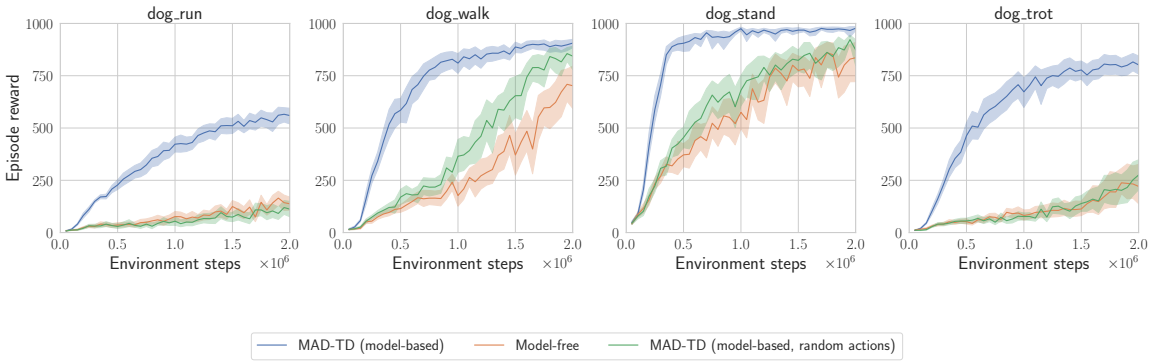

Figure 7.8: Return curves for the dog tasks when using on-policy, random and no model-generated data. When generating model-based data with random actions, performance of MAD-TD drops close to the model-free baseline, highlighting the importance of *on-policy* actions.

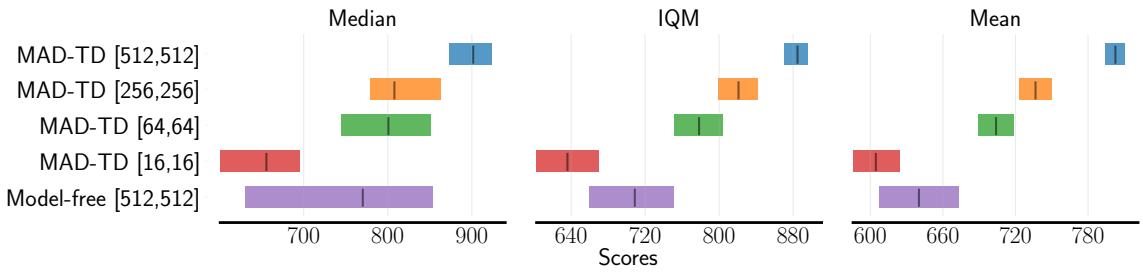

Figure 7.9: Performance evaluation when reducing the model size of the latent model in MAD-TD. The performance predictably drops with decreasing hidden layer size, however only strongly decreasing the model size below 64 reduces the performance below that of the model-free ablation.

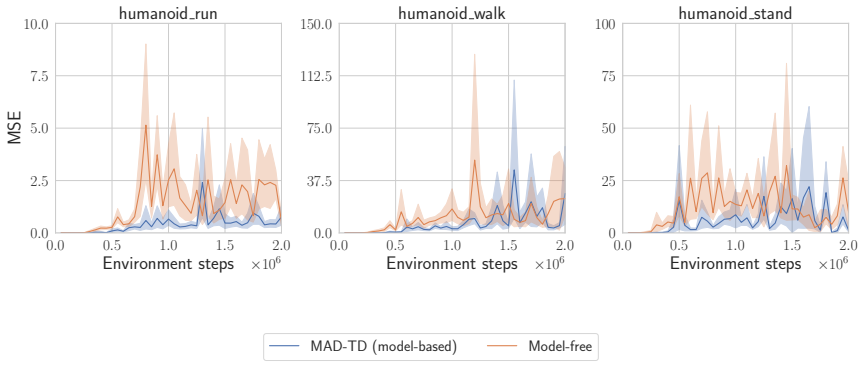

Figure 7.10: Magnitude of the difference between $Q(x, \pi(x))$ and $Q(x, \tilde{a})$, where $\tilde{a}$ is an adversarial perturbation of $\pi(x)$. Across the training why see higher perturbation magnitude for the runs without model correction data.

Chapter 8

Conclusion

Appendix A

Formal proofs and results

Move to main body where appropriate

A.1 Bound between Value-Aware Model Learning and VaGraM

The error in Taylor approximation $\mathcal{R}(V, s', f_\theta(s,a))$ is bounded by $\frac{M}{2}||s' - f_\theta(s,a)||^2$ with M dependent on the Hessian of the value function. Plugging this into the VAML loss and assuming worst case approximation errors, we obtain an upper bound on the VAML error:

$$
\begin{aligned}
&\mathbb{E}_{s',s,a\sim D}\left[\left(V\left(f_\theta(s,a)\right) - V(s'_0)\right)^2\right] \\
=&\mathbb{E}_{s',s,a\sim D}\left[\left((\nabla_s V(s)|_{s'})^\intercal (f_\theta(s,a) - s') + \mathcal{R}(V, s', f_\theta(s,a))\right)^2\right] \\
\leq&\mathbb{E}_{s',s,a\sim D}\left[\left(\left|(\nabla_s V(s)|_{s'})^\intercal (f_\theta(s,a) - s')\right| + \left|\mathcal{R}(V, s'_0, f_\theta(s,a))\right|\right)^2\right] \\
\leq&\mathbb{E}_{s',s,a\sim D}\left[\left(\left|(\nabla_s V(s)|_{s'})^\intercal (f_\theta(s,a) - s')\right| + \frac{M}{2}||s' - f_\theta(s,a)||^2\right)^2\right] \\
\leq&2\cdot\mathbb{E}_{s',s,a\sim D}\left[\left((\nabla_s V(s)|_{s'})^\intercal (f_\theta(s,a) - s')\right)^2\right] + 2\cdot\mathbb{E}_{s',s,a\sim D}\left[\frac{M^2}{4}||s' - f_\theta(s,a)||^4\right]
\end{aligned}
$$

Our experiments show that if we treat M like a tuneable hyperparameter, we obtain worse performance when optimizing this upper bound compared to VaGraM. The Hessian parameter is difficult to compute or estimate in practice and we find that most often, either the first or the second loss component will dominate when choosing heuristic values.

A.2 Analyzing the additional local minima of the Taylor approximation loss

We noted in the main paper that the direct Taylor approximation of the value function leads to a spurious local minimum. This is clear when looking at the loss for a single datapoint:

$$\begin{aligned}&\min_\theta \left((\nabla_s V(s)|_{s'})^\intercal f_\theta(s,a)\right)^2 \\ =&\min_\theta \left(\sum_{n=0}^{\dim(\mathcal{S})} (\nabla_s V(s)|_{s'})_n \cdot f_\theta(s,a)_n\right)^2\end{aligned}$$

Assuming that f is flexible and can predict any next state s' (i.e. by choosing $f = \theta$), the optimal solution is obtained from an undetermined linear system of equations. This system admits far more solutions than either the corresponding IterVAML loss or a mean squared error, and many of them will achieve arbitrary large value prediction errors. In fact, the equation describes a hyperplane of minimal solutions consisting of every weight vector that is orthogonal to the gradient of the value function at the reference sample, with $\dim(\mathcal{S}) - 1$ free variables. Therefore we need to enforce the closeness of the model prediction and the environment sample, since the Taylor approximation is only approximately valid in a close ball around the reference sample.

One way to achieve this closeness is by adding the second order Taylor term, which results in an additional MSE loss term. As pointed out in Section A.1, we did not achieve good performance when testing out this version, since it is difficult to compute the Hessian in higher state spaces and heuristically choosing a value as a hyperparameter proved to be difficult to tune in practice. Therefore, we approached the solution to this problem as outlined in the paper.

A.3 Proofs and mathematical clarifications

We provide the proofs for Subsection 7.3.1 in this section.

The first proposition relies on the existence of a deterministic mapping, which we prove here as a lemma. The second proposition requires a statement over the minimizers of an equation with a variance-like term, we prove a general result as a lemma.

The proof of the first lemma relies heavily on several propositions from Bertsekas and Shreve 1978, which are restated in Subsection A.3.3 for reader's convenience. Other topological statements are standard and can be find in textbooks such as **Munkres2018**.

Lemma 1 (Deterministic Representation Lemma)**.** *Let $\mathcal{X}$ be a compact, connected, metriz-*

able space. Let p be a continuous kernel from $\mathcal{X}$ to probability measures over $\mathcal{X}$. Let $\mathcal{Z}$ be a metrizable space. Consider a bijective latent mapping $\phi : \mathcal{X} \to \mathcal{Z}$ and any $V : \mathcal{Z} \to \mathbb{R}$. Assume that they are both continuous. Denote $V_{\mathcal{X}} = V \circ \phi$.

Then there exists a measurable function $f^* : \mathcal{Z} \to \mathcal{Z}$ such that we have $V(f^*(\phi(x))) = \mathbb{E}_p[V_{\mathcal{X}}(x')|x]$ for all $x \in \mathcal{X}$.

Proof: Since ϕ is a bijective continuous function over a compact space and maps to a Hausdorff space ($\mathcal{Z}$ is metrizable, which implies Hausdorff), it is a homeomorphism. The image of $\mathcal{X}$ under ϕ, $\mathcal{Z}_{\mathcal{X}}$ is then connected and compact. Since $\mathcal{X}$ is metrizable and compact and ϕ is a homeomorphism, $\mathcal{Z}_{\mathcal{X}}$ is metrizable and compact. Let $\theta_{V,\mathcal{X}}(x) = \mathbb{E}_{x' \sim p(\cdot|x)}[V(x')]$. Then, $\theta_{V,\mathcal{X}}$ is continuous (Proposition Proposition 3). Define $\theta_{V,\mathcal{X}} = \theta_{V,\mathcal{Z}} \circ \phi$. Since ϕ is a homeomorphism, ϕ^{-1} is continuous. The function $\theta_{V,\mathcal{Z}}$ can be represented as a composition of continuous functions $\theta_{V,\mathcal{Z}} = \theta_{V,\mathcal{X}} \circ \phi^{-1}$ and is therefore continuous.

As $\mathcal{Z}_{\mathcal{X}}$ is compact, the continuous function V takes a maximum and minimum over the set $\mathcal{Z}_{\mathcal{X}}$. This follows from the compactness of $\mathcal{Z}_{\mathcal{X}}$ and the extreme value theorem. Furthermore $V_{\min} \leq \theta_{V,\mathcal{Z}}(z) \leq V_{\max}$ for every $z \in \mathcal{Z}_{\mathcal{X}}$. By the intermediate value theorem over compact, connected spaces, and the continuity of V, for every value $V_{\min} \leq v \leq V_{\max}$, there exists a $z \in \mathcal{Z}_{\mathcal{X}}$ so that $V(z) = v$.

Let $h : \mathcal{Z}_{\mathcal{X}} \times \mathcal{Z}_{\mathcal{X}} \to \mathbb{R}$ be the function $h(z, z') = |\theta_{V,\mathcal{Z}}(z) - V(z')|^2$. As h is a composition of continuous functions, it is itself continuous. Let $h^*(z) = \min_{z' \in \mathcal{Z}_{\mathcal{X}}} h(z, z')$. For any $z \in \mathcal{Z}_{\mathcal{X}}$, by the intermediate value argument, there exist z' such that $V(z') = v$. Therefore $h^*(z)$ can be minimized perfectly for all $z \in \mathcal{Z}_{\mathcal{X}}$.

Since $\mathcal{Z}_{\mathcal{X}}$ is compact, h is defined over a compact subset of $\mathcal{Z}$. By Proposition Proposition 4, there exists a measurable function $f^*(z)$ so that $\min_{z'} h(z, z') = h(z, f^*(z)) = 0$. Therefore, the function f^* has the property that $V(f^*(z)) = \mathbb{E}_p[V(z')|z]$, as this minimizes the function h.

Now consider any $x \in \mathcal{X}$ and its corresponding $z = \phi(x)$. As $h(z, f^*(z)) = |\theta_{V,\mathcal{Z}}(z) - V(f^*(z))|^2 = 0$ for any $z \in \mathcal{Z}_{\mathcal{X}}$, $V(f^*(\phi(x))) = \theta_{v,\mathcal{Z}}(z) = \mathbb{E}_p[V_{\mathcal{X}}(x')|x]$ as desired.

□

The following lemma shows that a function that minimizes a quadratic and a variance term cannot be the minimum function of the quadratic. This is used to show that the minimum of the MuZero value function learning term is not the same as applying the model-based Bellman operator.

Lemma 2. *Let $g : \mathcal{X} \to \mathbb{R}$ be a function that is not constant almost everywhere and let μ be a non-degenerate probability distribution over $\mathcal{X}$. Let $\mathcal{F}$ be an open function space with $g \in \mathcal{F}$. Let $\mathcal{L}(f) = \mathbb{E}_{x \sim \mu}\left[(f(x) - g(x))^2\right] + \mathbb{E}_{x \sim \mu}[f(x)g(x)] - \mathbb{E}_{x \sim \mu}[f(x)]\,\mathbb{E}_{\mu}[g(x)]$. Then*

$g \notin \arg\min_{f \in \mathcal{F}} \mathcal{L}(f)$.

Proof: The proof follows by showing that there is a descent direction from g that improves upon $\mathcal{L}$. For this, we construct the auxiliary function $\hat{g}(x) = g(x) - \epsilon g(x)$. Substituting $\hat{g}$ into $\mathcal{L}$ yields

$$\begin{aligned} &\epsilon^2 \mathbb{E}_\mu \left[g(x)^2\right] + \mathbb{E}_\mu \left[\left(g(x) - \epsilon g(x)\right) g(x)\right] - \mathbb{E}_\mu \left[\left(g(x) - \epsilon g(x)\right)\right] \mathbb{E}_\mu \left[g(x)\right] \\ &= \epsilon^2 \mathbb{E}_\mu \left[g(x)^2\right] + (1-\epsilon)\, \mathbb{E}_\mu \left[g(x)^2\right] - (1-\epsilon)\, \mathbb{E}_\mu \left[g(x)\right]^2 . \end{aligned}$$

Taking the derivative of this function wrt to ϵ yields

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}\epsilon} \epsilon^2 \mathbb{E}_\mu \left[g(x)^2\right] + (1-\epsilon)\, \mathbb{E}_\mu \left[g(x)^2\right] - (1-\epsilon)\, \mathbb{E}_\mu \left[g(x)\right]^2 \\ &= 2\epsilon\, \mathbb{E}_\mu \left[g(x)^2\right] - \mathbb{E}_\mu \left[g(x)^2\right] + \mathbb{E}_\mu \left[g(x)\right]^2 . \end{aligned}$$

Setting ϵ to 0, obtain

$$\mathbb{E}_\mu \left[g(x)\right]^2 - \mathbb{E}_\mu \left[g(x)^2\right] = \mathrm{Var}_\mu \left[g(x)\right]$$

By the Cauchy-Schwartz inequality, the variance is only 0 for a $g(x)$ constant almost everywhere. However, this violates the assumption. Therefore for any $\epsilon > 0$ $\mathcal{L}\left(\hat{g}\right) \leq \mathcal{L}\left(g\right)$, due to the descent direction given by $-g(x)$. As we assume that the function class is open, there also exists an $\epsilon > 0$ for which $g(x) - \epsilon g(x) \in \mathcal{F}$.

□

A.3.1 Main propositions

Proposition 1. *Let M be an MDP with a compact, connected state space $\mathcal{X} \subseteq \mathcal{Y}$, where $\mathcal{Y}$ is a metrizable space. Let the transition kernel p be continuous. Let $\mathcal{Z}$ be a metrizable space. Consider a bijective latent mapping $\phi : \mathcal{Y} \to \mathcal{Z}$ and any value function approximation $V : \mathcal{Z} \to \mathbb{R}$. Assume that they are both continuous. Denote $V_\mathcal{X} = V \circ \phi$.*

Then there exists a measurable function $f^* : \mathcal{Z} \to \mathcal{Z}$ such that we have $V(f^*(\phi(x))) = \mathbb{E}_p\left[V_\mathcal{X}(x')|x\right]$ for all $x \in \mathcal{X}$.

Furthermore, the same f^* is a minimizer of the population IterVAML loss:

$$f^* \in \arg\min_{\hat{f}} \mathbb{E}\left[\hat{\mathcal{L}}_{IterVAML}(\hat{f}; V_\mathcal{X})\right].$$

Proof: The existence of f^* follows under the stated assumptions (compact, connected and metrizable state space, metrizable latent space, continuity of all involved functions) from Lemma Lemma 1.

The rest of the proof follows standard approaches in textbooks such as **Gyrfi2002ADT**. First, expand the equation to obtain:

$$\begin{aligned}\mathbb{E}\left[\hat{\mathcal{L}}^n_{\text{IterVAML}}(f; V_{\mathcal{X}}, \mathcal{D})\right] &= \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\left[V\left(f\left(\phi(x_i)\right)\right) - V_{\mathcal{X}}(x'_i)\right]^2\right] \\ &= \mathbb{E}\left[\left[V\left(f\left(\phi(x)\right)\right) - \mathbb{E}\left[V_{\mathcal{X}}(x')\middle|x\right] + \mathbb{E}\left[V_{\mathcal{X}}(x')\middle|x\right] - V_{\mathcal{X}}(x')\right]^2\right].\end{aligned}$$

After expanding the square, we obtain three terms:

$$\begin{aligned}&\mathbb{E}\left[\left|V\left(f\left(\phi(x)\right)\right) - \mathbb{E}\left[V_{\mathcal{X}}(x')\middle|x\right]\right|^2\right] \\ &+2\mathbb{E}\left[\left[V\left(f\left(\phi(x)\right)\right) - \mathbb{E}\left[V_{\mathcal{X}}(x')\middle|x\right]\right]\left[\mathbb{E}\left[V_{\mathcal{X}}(x')\middle|x\right] - V_{\mathcal{X}}(x')\right]\right] \\ &+\mathbb{E}\left[\left|\mathbb{E}\left[V_{\mathcal{X}}(x')\middle|x\right] - V_{\mathcal{X}}(x')\right|^2\right]\end{aligned}$$

Apply the tower property to the inner term to obtain:

$$\begin{aligned}&2\mathbb{E}\left[\left[V\left(f\left(\phi(x)\right)\right) - \mathbb{E}\left[V_{\mathcal{X}}(x')\middle|x\right]\right]\left[\mathbb{E}\left[V_{\mathcal{X}}(x')\middle|x\right] - V_{\mathcal{X}}(x')\right]\right] \\ =&2\mathbb{E}\left[\left[V\left(f\left(\phi(x)\right)\right) - \mathbb{E}\left[V_{\mathcal{X}}(x')\middle|x\right]\right]\underbrace{\mathbb{E}\left[\mathbb{E}\left[V_{\mathcal{X}}(x')\middle|x\right] - V_{\mathcal{X}}(x')\middle|x'\right]}_{=0}\right] = 0.\end{aligned}$$

Since the statement we are proving only applies to the minimum of the IterVAML loss, we will work with the arg min of the loss function above. The resulting equation contains a term dependent on f and one independent of f:

$$\begin{aligned}&\arg\min_f \mathbb{E}\left[\left|V\left(f\left(\phi(x)\right)\right) - \mathbb{E}\left[V_{\mathcal{X}}(x')\middle|x\right]\right|^2\right] + \mathbb{E}\left[\left|\mathbb{E}\left[V_{\mathcal{X}}(x')\middle|x\right] - V_{\mathcal{X}}(x')\right|^2\right] \\ =\ &\arg\min_f \mathbb{E}\left[\left|V\left(f\left(\phi(x)\right)\right) - \mathbb{E}\left[V_{\mathcal{X}}(x')\middle|x\right]\right|^2\right].\end{aligned}$$

Finally, it is easy to notice that $V\left(f^*\left(\phi(x)\right)\right) = \mathbb{E}\left[V_{\mathcal{X}}(x')|x\right]$ by the definition of f^*. Therefore f^* minimizes the final loss term and, due to that, the IterVAML loss.

□

Proposition 2. *Assume a non-deterministic MDP with a fixed, but arbitrary policy π, and let p be the transition kernel. Let $\mathcal{V}$ be an open set of functions, and assume that it is*

Bellman complete: $\forall V \in \mathcal{V} : \mathcal{T}V \in \mathcal{V}$.

Then for any $V' \in \mathcal{V}$ *that is not a constant function,* $\mathcal{T}V' \notin \arg\min_{\hat{V}\in\mathcal{V}} \mathbb{E}_{\mathcal{D}}\left[\hat{\mathcal{L}}^1_{MuZero}(p, \hat{V}; \mathcal{D}, V')\right]$.

Notation: For clarity of presentation denote samples from the real environment as $x^{(n)}$ for the n-th sample after a starting point $x^{(0)}$. This means that $x^{(n+1)}$ is drawn from $p\left(\cdot\middle|x^{(n)}\right)$. Similarly, $\hat{x}^{(n)}$ is the n-th sample drawn from the model, with $\hat{x}^{(0)} = x^{(0)}$. All expectations are taken over $x_i^{(0)} \sim \mu$ where μ is the data distribution, $\hat{x}_i^{(1)} \sim \hat{p}\left(\cdot\middle|x_i^{(0)}\right)$, $x_i^{(1)} \sim p\left(\cdot\middle|x_i^{(0)}\right)$, and $x_i^{(2)} \sim p\left(\cdot\middle|x_i^{(1)}\right)$. We use the tower property several times, all expectations are conditioned on ${x_i^{(0)}}_i$.

Proof: By assumption, let $\hat{p}$ in the MuZero loss be the true transition kernel p. Expand the MuZero loss by $\left[r\left(x_i^{(1)}\right) + \gamma V'\left(x_i^{(2)}\right)\right]$ and take its expectation:

$$
\begin{aligned}
&\mathbb{E}\left[\hat{\mathcal{L}}^1_{\text{MuZero}}(\hat{p}, \hat{V}; \mathcal{D}, V')\right] \\
=&\mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N}\left[\hat{V}\left(\hat{x}_i^{(1)}\right) - \left[r\left(x_i^{(1)}\right) + \gamma V'\left(x_i^{(2)}\right)\right]\right]^2\right] \\
=&\,\mathbb{E}\left[\left[\hat{V}\left(\hat{x}_i^{(1)}\right) - (\mathcal{T}V')\left(\hat{x}_i^{(1)}\right) + (\mathcal{T}V')\left(\hat{x}_i^{(1)}\right) - \left[r\left(x_i^{(1)}\right) + \gamma V'\left(x_i^{(2)}\right)\right]\right]^2\right] \\
=&\,\mathbb{E}\left[\left(\hat{V}\left(\hat{x}_i^{(1)}\right) - (\mathcal{T}V')\left(\hat{x}_i^{(1)}\right)\right)^2\right] + \\
&\,2\,\mathbb{E}\left[\left(\hat{V}\left(\hat{x}_i^{(1)}\right) - (\mathcal{T}V')\left(\hat{x}_i^{(1)}\right)\right)\left((\mathcal{T}V')\left(\hat{x}_i^{(1)}\right) - \left[r\left(x_i^{(1)}\right) + \gamma V'\left(x_i^{(2)}\right)\right]\right)\right] + \\
&\,\mathbb{E}\left[\left((\mathcal{T}V')\left(\hat{x}_i^{(1)}\right) - \left[r\left(x_i^{(1)}\right) + \gamma V'\left(x_i^{(2)}\right)\right]\right)^2\right]
\end{aligned}
$$

We aim to study the minimizer of this term. The first term (Section A.3.1) is the regular bootstrapped Bellman residual with a target V'. The third term (Section A.3.1) is independent of $\hat{V}$, so we can drop it when analyzing the minimization problem.

The second term (Section A.3.1) simplifies to

$$
\mathbb{E}\left[\hat{V}\left(\hat{x}_i^{(1)}\right)\left((\mathcal{T}V')\left(\hat{x}_i^{(1)}\right) - \left[r\left(x_i^{(1)}\right) + \gamma V'\left(x_i^{(2)}\right)\right]\right)\right]
$$

as the remainder is independent of $\hat{V}$ again.

This remaining term however is not independent of $\hat{V}$ and not equal to 0 either. Instead, it decomposes into a variance-like term, using the conditional independence of $\hat{x}_i^{(1)}$ and $x_i^{(1)}$

given $x_i^{(0)}$:

$$\begin{aligned}
&\mathbb{E}\left[\hat{V}\left(\hat{x}_i^{(1)}\right)\left(\left(\mathcal{T}V'\right)\left(\hat{x}_i^{(1)}\right)-\left[r\left(x_i^{(1)}\right)+\gamma V'\left(x_i^{(2)}\right)\right]\right)\right]\\
=&\,\mathbb{E}\left[\hat{V}\left(\hat{x}_i^{(1)}\right)\left(\mathcal{T}V'\right)\left(\hat{x}_i^{(1)}\right)\right]-\mathbb{E}\left[\hat{V}\left(\hat{x}_i^{(1)}\right)\left[r\left(x_i^{(1)}\right)+\gamma V'\left(x_i^{(2)}\right)\right]\right]\\
=&\,\mathbb{E}\left[\hat{V}\left(\hat{x}_i^{(1)}\right)\left(\mathcal{T}V'\right)\left(\hat{x}_i^{(1)}\right)\right]-\mathbb{E}\left[\hat{V}\left(\hat{x}_i^{(1)}\right)\right]\mathbb{E}\left[\left[r\left(x_i^{(1)}\right)+\gamma V'\left(x_i^{(2)}\right)\right]\right].
\end{aligned}$$

Combining this with Section A.3.1, we obtain

$$\begin{aligned}
&\mathbb{E}\left[\hat{\mathcal{L}}^1_{\text{MuZero}}(p,\hat{V};\mathcal{D},V')\right]\\
=&\,\mathbb{E}\left[\left(\hat{V}\left(\hat{x}_i^{(1)}\right)-\left(\mathcal{T}V'\right)\left(\hat{x}_i^{(1)}\right)\right)^2\right]+\\
&\,\mathbb{E}\left[\hat{V}\left(\hat{x}_i^{(1)}\right)\left(\mathcal{T}V'\right)\left(\hat{x}_i^{(1)}\right)\right]-\mathbb{E}\left[\hat{V}\left(\hat{x}_i^{(1)}\right)\right]\mathbb{E}\left[\left[r\left(x_i^{(1)}\right)+\gamma V'\left(x_i^{(2)}\right)\right]\right].
\end{aligned}$$

The first summand is the Bellman residual, which is minimized by $\mathcal{T}V'$, which is in the function class by assumption. However, by Lemma 2, $\mathcal{T}V'$ does not minimize the whole loss term under the conditions (open function class, non-constant value functions, and non-degenerate transition kernel).

□

Discussion: The proof uses Bellman completeness, which is generally a strong assumption. However, this is only used to simplify showing the contradiction at the end, removing it does not remove the problems with the loss. The proof of Lemma 2 can be adapted to the case where $f(x)$ minimizes the difference to $g(x)$, instead of using $g(x)$ as the global minimum, but some further technical assumptions about the existence of minimizers and boundary conditions are needed. The purpose here is to show that even with very favorable assumptions such as Bellman completeness, the MuZero value function learning algorithm will not converge to an expected solution.

Similarly, the condition of openness of the function class simply ensures that there exists a function "nearby" that minimizes the loss better. This is mostly to remove edge cases, such as the case where the function class exactly contains the correct solution. Such cases, while mathematically valid, are uninteresting from the perspective of learning functions with flexible function approximations.

We only show the proof for the single step version and remove action dependence to remove notational clutter, the action-dependent and multi-step versions follow naturally.

A.3.2 Comparison of IterVAML and MuZero for model learning

If MuZero is instead used to only update the model $\hat{p}$, we obtain a similarity between MuZero and IterVAML. This result is similar to the one presented in Grimm, André Barreto, et al. 2021, so we only present it for completeness sake, and not claim it as a fully novel contribution of our paper. While Grimm, André Barreto, et al. 2021 show that the whole MuZero loss is an upper bound on an IterVAML-like term, we highlight the exact term the model learning component minimizes. However, we think it is still a useful derivation as it highlights some of the intuitive similarities and differences between IterVAML and MuZero and shows that they exist as algorithms on a spectrum spanned by different estimates of the target value function.

We will choose a slightly different expansion than before, using $\mathbb{E}_{x_i^{(1)}, x_i^{(2)} \sim p}\left[\left[r\left(x_i^{(1)}\right) + \gamma V'\left(x_i^{(2)}\right)\right]\right] = \mathbb{E}\left[\mathcal{T}V'\left(x_i^{(1)}\right)\right]$

$$
\begin{aligned}
&\mathbb{E}\left[\hat{\mathcal{L}}^1_{\text{MuZero}}(\hat{p}, V; \mathcal{D}, V')\right] \\
&= \mathbb{E}\left[\left[V\left(\hat{x}_i^{(1)}\right) - \mathbb{E}\left[\mathcal{T}V'\left(x_i^{(1)}\right)\right] + \mathbb{E}\left[\mathcal{T}V'\left(x_i^{(1)}\right)\right] - \left[r\left(x_i^{(1)}\right) + \gamma V'\left(x_i^{(2)}\right)\right]\right]^2\right] \\
&= \mathbb{E}\left[\left(V\left(\hat{x}_i^{(1)}\right) - \mathbb{E}\left[\mathcal{T}V'\left(x_i^{(1)}\right)\right]\right)^2\right] + \\
&\quad 2\mathbb{E}\left[\left(V\left(\hat{x}_i^{(1)}\right) - \mathbb{E}\left[\mathcal{T}V'\left(x_i^{(1)}\right)\right]\right)\left(\mathbb{E}\left[\mathcal{T}V'\left(x_i^{(1)}\right)\right] - \left[r\left(x_i^{(1)}\right) + \gamma V'\left(x_i^{(2)}\right)\right]\right)\right] + \\
&\quad \mathbb{E}\left[\left(\mathbb{E}\left[\mathcal{T}V'\left(x_i^{(1)}\right)\right] - \left[r\left(x_i^{(1)}\right) + \gamma V'\left(x_i^{(2)}\right)\right]\right)^2\right].
\end{aligned}
$$

The first summand (Subsection A.3.2) is similar to the IterVAML loss, instead of using the next state's value function, the one-step bootstrap estimate of the Bellman operator is used.

The third term (Subsection A.3.2) is independent of $\hat{p}$ and can therefore be dropped. The second term decomposes into two terms again,

$$
\begin{aligned}
&\mathbb{E}\left[\left(V\left(\hat{x}_i^{(1)}\right) - \mathbb{E}\left[\mathcal{T}V'\left(x_i^{(1)}\right)\right]\right)\left(\mathbb{E}\left[\mathcal{T}V'\left(x_i^{(1)}\right)\right] - \left[r\left(x_i^{(1)}\right) + \gamma V'\left(x_i^{(2)}\right)\right]\right)\right] \\
&= \mathbb{E}\left[\left(V\left(\hat{x}_i^{(1)}\right)\right)\left(\mathbb{E}\left[\mathcal{T}V'\left(x_i^{(1)}\right)\right] - \left[r\left(x_i^{(1)}\right) + \gamma V'\left(x_i^{(2)}\right)\right]\right)\right] - \\
&\quad \mathbb{E}\left[\mathbb{E}\left[\mathcal{T}V'\left(x_i^{(1)}\right)\right]\left(\mathbb{E}\left[\mathcal{T}V'\left(x_i^{(1)}\right)\right] - \left[r\left(x_i^{(1)}\right) + \gamma V'\left(x_i^{(2)}\right)\right]\right)\right].
\end{aligned}
$$

The first summand is equal to 0, due to the conditional independence of $\hat{x}_i^{(1)}$ and $x_i^{(1)}$,

$$\begin{aligned}
&\mathbb{E}\left[\left(V\left(\hat{x}_i^{(1)}\right)\right)\left(\mathbb{E}\left[\mathcal{T}V'\left(x_i^{(1)}\right)\right] - \left[r\left(x_i^{(1)}\right) + \gamma V'\left(x_i^{(2)}\right)\right]\right)\right] \\
&= \mathbb{E}\left[V\left(\hat{x}_i^{(1)}\right)\right] \underbrace{\left(\mathbb{E}\left[\mathbb{E}\left[\mathcal{T}V'\left(x_i^{(1)}\right)\right]\right] - \mathbb{E}\left[\left[r\left(x_i^{(1)}\right) + \gamma V'\left(x_i^{(2)}\right)\right]\right]\right)}_{=0} = 0.
\end{aligned}$$

The second remaining summand is independent of $\hat{p}$ and therefore irrelevant to the minimization problem.

Therefore, the MuZero model learning loss minimizes

$$\mathbb{E}\left[\hat{\mathcal{L}}^1_{\text{MuZero}}(\hat{p}, V; \mathcal{D}, V')\right] = \mathbb{E}\left[\left(V\left(\hat{x}_i^{(1)}\right) - \mathbb{E}\left[\mathcal{T}V'\left(x_i^{(1)}\right)\right]\right)^2\right].$$

In conclusion, the MuZero loss optimizes a closely related function to the IterVAML loss when used solely to update the model. There are three differences: First, the bootstrap value function estimator is used instead of the value function as the target value. Second, the current value function estimate is used for the model sample and the target network (if used) is applied for the bootstrap estimate. If the target network is equal to the value function estimate, this difference disappears. Finally, the loss does not contain the inner expectation around the model value function. This can easily be added to the loss and its omission in MuZero is unsurprising, as the loss was designed for deterministic environments and models.

The similarity between the losses suggests a potential family of decision-aware algorithms with different bias-variance characteristics, of which MuZero and IterVAML can be seen as two instances. It is also interesting to note that even without updating the value function, the MuZero loss performs an implicit minimization of the difference between the current value estimate and the Bellman operator via the model prediction. This is an avenue for further research, as it might explain some of the empirical success of the method disentangled from the value function update.

A.3.3 Propositions from Bertsekas and Shreve 1978

For convenience, we quote some results from Bertsekas and Shreve 1978. These are used in the proof of Lemma 1.

Proposition 3 (Proposition 7.30 of Bertsekas and Shreve 1978)**.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be separable metrizable spaces and let $q(\mathrm{d}y|x)$ be a continuous stochastic kernel on $\mathcal{Y}$ given $\mathcal{X}$. If*

$f \in \mathcal{C}(\mathcal{X} \times \mathcal{Y})$, the function $\lambda : \mathcal{X} \to \mathbb{R}$ defined by

$$\lambda(x) = \int f(x, y) q(\mathrm{d}y|x)$$

is continuous.

Proposition 4 (Proposition 7.33 of Bertsekas and Shreve 1978)**.** *Let $\mathcal{X}$ be a metrizable space, $\mathcal{Y}$ a compact metrizable space, $\mathcal{D}$ a closed subset of $\mathcal{X} \times \mathcal{Y}$, $\mathcal{D}_x = \{y|(x, y) \in \mathcal{D}\}$, and let $f : \mathcal{D} \to \mathbb{R}^*$ be lower semicontinuous. Let $f^* : proj_{\mathcal{X}}(\mathcal{D}) \to \mathbb{R}^*$ be given by*

$$f^*(x) = \min_{y \in \mathcal{D}_x} f(x, y).$$

Then $proj_{\mathcal{X}}(\mathcal{D})$ is closed in $\mathcal{X}$, f^* is lower semicontinuous, and there exists a Borel-measurable function $\phi : proj_{\mathcal{X}}(\mathcal{D}) \to \mathcal{Y}$ such that range$(\phi) \subset \mathcal{D}$ and

$$f\left[x, \phi(x)\right] = f^*(x), \quad \forall x \in proj_{\mathcal{X}}(\mathcal{D}).$$

In our proof, we construct f^* as the minimum of an IterVAML style loss and equate ϕ with the function we call f in our proof. The change in notation is chosen to reflect the modern notation in MBRL – in the textbook, older notation is used.

A.3.4 Bias due to the stabilizing loss

As highlighted in Subsection 6.2.1, the addition of a stabilizing loss is necessary to achieve good performance with any of the loss functions. With deterministic models, the combination $\hat{\mathcal{L}}_{\text{IterVAML}} + \hat{\mathcal{L}}^n_{\text{latent}}$ is stable, but the conditions for recovering the optimal model are not met anymore. This is due to the fact that $\arg\min_{\hat{f}} \mathbb{E}\left[\hat{\mathcal{L}_{\text{latent}}}(\hat{f}, \phi; \mathcal{D})\right] = \mathbb{E}\left[\phi(x'\right]$, but in general $\mathbb{E}\left[V(\phi(x'))\right] \neq V\left(\mathbb{E}\left[\phi(x')\right]\right)$. While another stabilization technique could be found that does not have this problem, we leave this for future work.

A.3.5 Multi-step IterVAML

In Subsection 6.1.1 the multi-step extension of IterVAML is introduced. As MVE and SVG require multi-step rollouts to be effective, it became apparent that simply forcing the one-step prediction of the value function to be correct is insufficient to obtain good performance. We therefore extended the loss into a multi-step variant.

Using linear algebra notation for simplicity, the single-step IterVAML loss enforces

$$\min \left| \langle V, P(x, a) - \hat{P}(x, a) \rangle \right|^2$$

The n-step variant then seeks to enforce a minimum between n applications of the respective

	Model loss	Value est.	Policy est.	Actor policy	DAML	Latent
λ-IterVAML	BYOL	MVE	SVG	direct	✓	✓
λ-MuZero	BYOL	MuZero	SVG	direct	✓	✓
MuZero (Schrittwieser et al. 2020)	-	MuZero	-	MCTS	✓	✓
Eff.-MuZero (Ye et al. 2021)	-	MuZero	-	MCTS	✓	✓
ALM (Ghugare et al. 2023)	ALM-ELBO	m.-free	ALM-SVG	direct	✓	✓
TD-MPC (N. Hansen, X. Wang, and Su 2022)	BYOL	m.-free	DDPG	MPC	✓	✓
MBPO (Janner et al. 2019a)	MLE	SAC	SAC	direct	-	-
Dreamer (**Hafner2020Dream**)	ELBO	MVE	SVG	direct	-	✓
IterVAML (Farahmand 2018)	-	Dyna	-	direct	✓	-
VaGraM (Voelcker, Liao, et al. 2022)	weigthed MLE	SAC	SAC	direct	✓(?)	-

Table A.1: An overview of different model-based RL algorithms and how they fit into the λ-AC family. The first two are the empirical algorithms tested in this work. The next section contains work that falls well into the λ-AC family as described in this paper. The similarities between these highlight that further algorithms are easily constructed, i.e. ALM (Ghugare et al. 2023) combined with MPC. The final section contains both popular algorithms and closely related work which inform the classification, but are not part of it since they are either not latent or not decision-aware.

transition operators

$$\min \left| \langle V, P^n(x,a) - \hat{P}^n(x,a) \rangle \right|^2$$

The sample-based variant is a proper regression target, as $V(x^{(j)}$ is an unbiased sample from $P^n(x,a)$. It is easy to show following the same techniques as used in the proofs of propositions 1 and 2 that the sample-based version indeed minimizes the IterVAML loss in expectation.

Finally, we simply sum over intermediate n-step versions which results in the network being forced to learn a compromise between single-step and multi-step fidelity to the value function.

A.4 λ-Reinforcement Learning Algorithms

We introduce the idea of the λ-AC framework in Section 6.2. The characterization of the family is fairly broad, and it contains more algorithms than those directly discussed in this paper. An overview of related algorithms is presented in Table A.1. However, we found it useful to establish that many recently proposed algorithms are closely related and contain components that can be combined freely with one another. While the community treats, i.e. MuZero, as a fixed, well-defined algorithm, it might be more useful to treat it as a certain set of implementation and design decisions under an umbrella of related algorithms. Given the findings of this paper for example, it is feasible to evaluate a MuZero alternative which simply replaces the value function learning algorithm with MVE, which should be

more robust in stochastic environments.

A full benchmarking and comparison effort is out of scope for this work. However, we believe that a more integrative and holistic view over the many related algorithms in this family is useful for the community, which is why we present it here.

A.5 Motivating examples

A.5.1 Motivating example for observation spaces

To further motivate our focus on observation models, consider the common problem of representing rotational angles. As these are continuous values, discretized one-hot representations may introduce errors that may harm efficient control. When choosing a continuous representation, designers are faced with the choice between representing the angle as $\omega \in [-\pi, \pi)$ (the exclusion of the right endpoint is arbitrary), or by a decomposition into $[\sin(\omega), \cos(\omega)] \in [0, 1]^2$. The choice of representation has measurable impact on the ease of learning: the former is 1D instead of 2D, which can reduce the size of networks needed when dealing with many angles. However a complication with the first representation is it is discontinuous at the right endpoint, in the sense that $\lim_{x\to\pi} x = -\pi$ (this is generally due to the structure of $\mathbb{R}/2\pi\mathbb{Z}$). This creates a peculiar continuity condition for functions on this representation space: for a function f to be continuous, it must be continuous and also satisfy the boundary condition $\lim_{x\to\pi} f(x) = f(-\pi)$. Without explicitly enforcing this, the majority of functions learned on this domain (such as estimated value functions) will be discontinous, leading to additional difficulties in the learning process. On the other hand, this issue does not exist for the second representation, potentially making learning much easier.

The important question is that of *similarity* and *continuity*: in our pendulum example, the more compact representation breaks the intuitive notion that states that behave similariy should be mapped to close representations.

A.5.2 Distractions

The optimality of features can be measured in how close the projection of the true value function onto their span is to the ground truth, i.e. in the L_2 norm. This raises the question under what conditions the top k eigenvectors or singular vectors would not span the value function well. For this, we turn to our notion of an MDP with distractions.

Proposition 5 (Suboptimality of top k eigenspaces with distractions)**.** *Assume an MDP with distraction composed of two independent processes $\mathcal{M}$ and $\mathcal{N}$ according to Definition 11. Let $v_1, \ldots, v_n$ and $u_1, \ldots, u_m$ be the eigenvectors of $\mathcal{M}$ and $\mathcal{N}$ respectively, with associated real eigenvalues λ_j and μ_i ordered. Assume $\forall i < k : \mu_i > \lambda_2$ and $r_{\mathcal{N}} = 0$. Let U_k be an*

orthonormal basis for the top-k eigenspace of $\mathcal{M}\otimes\mathcal{N}$. *Then* $\text{span}(U_k) = \text{span}(\{\mathbf{1}\otimes v_i | \forall i \leq k\})$ *and* $\text{proj}_{U_k}(r_\mathcal{M} \otimes \mathbf{1}) = (\sum_{i=0}^{m} r_i/n)\mathbf{1}_{n\cdot m}$.

This means that there is a natural notion of a process in which the top-k eigenvectors do not span the reward function well. In this case, the distracting process has larger eigenvalues than the reward-relevant process. As the eigenvector basis is composed of Kronecker products of the individual eigenvectors, the top-k eigenspace contains redundant copies of the reward relevant processes eigenvectors. By Lemma 7, this directly implies suboptimal value function approximation. Note that our assumptions here are restrictive as we consider a *worst case* distraction for clarity, but the problem emerges whenever the distracting process has several large eigenvalues.

A.6 Helpful definitions and lemmata

This section contains helpful lemmata that are used for our proofs. Where we took these from existing work, we provide references, otherwise the proofs are our own, although probably also known in the literature.

A.6.1 Linear Algebra

As before, for matrices V we write $\text{span}(V)$ to denote the span of their column vectors.

Definition 12 (Top-k singular vectors)**.** *Let* $P^\pi = U^\top \Sigma V$ *be the singular value decomposition, and assume that the diagonal of* Σ *is arranged in decreasing order. The first* k *rows of* $U^\top$ *and the first* k *columns of* V *are called the top-k left and right singular vectors, respectively.*

Lemma 3 (Spectrum of Kronecker product matrix)**.** *Consider two non-singular matrices* M *and* N. *Let* λ_i *be the eigenvalues of* M *and* μ_j *be the eigenvalues of* N, *with eigenvectors* u_i *and* v_j *respectively. Then the eigenvalues of* $M \otimes N$ *are* $\lambda_i\mu_j$ *with eigenvectors* $u_i \otimes v_j$ *respectively.*

Proof. For any eigenvector u_i of M and v_j of N, we have

$$(M \otimes N)(u_i \otimes v_j) = (Mu_i) \otimes (Nv_j) = (\lambda_i u_i) \otimes (\mu_j v_j) = \lambda_i\mu_j (u_i \otimes v_j).$$

So $u_i \otimes v_j$ is an eigenvector of $M \otimes N$ with eigenvalue $\lambda_i\mu_j$ □

Lemma 4 (Orthogonalized bases and the Kronecker product)**.** *Let* $V \in \mathbb{R}^{n\times m}$ *be a matrix. Let* $\text{orth}(V)$ *be a matrix of any orthonormal basis vectors for the column vectors of* V. *Then* $\text{span}(V \otimes \mathbf{1}) = \text{span}(\text{orth}(V) \otimes \mathbf{1})$. *Furthermore* $(1/k)\text{orth}(V) \otimes \mathbf{1}$ *is an orthonormal basis of* $\text{span}(V \otimes \mathbf{1})$.

Proof. Let $V_i \otimes \mathbf{1}^k$ be any column vector of $V \otimes \mathbf{1}^k$. Let $\alpha_1, \ldots, \alpha_m$ be coefficients so that $V_i = \sum_{j=1}^m \alpha_j \text{orth}(V)_j$. Then

$$\sum_{j=1}^{m} \alpha_j \left(\text{orth}(V)_j \otimes \mathbf{1}^k\right) = \left(\sum_{j=1}^{m} \alpha_j \text{orth}(V)_j\right) \otimes \mathbf{1}^k = V_i \otimes \mathbf{1}^k,$$

following the standard associative and distributive properties of the Kronecker product.

As every vector in the original span can be represented in the orthogonalized span, the two are equivalent.

Finally note that

$$\frac{1}{k} \left(\text{orth}(V) \otimes \mathbf{1}\right)_i^\top \left(\text{orth}(V) \otimes \mathbf{1}\right)_i = \frac{1}{k} \sum_{j=1}^{k} \left(\text{orth}(V)_i^\top \text{orth}(V)_i^\top\right) = 1, \text{ and}$$

$$\frac{1}{k} \left(\text{orth}(V) \otimes \mathbf{1}\right)_j^\top \left(\text{orth}(V) \otimes \mathbf{1}\right)_i = \frac{1}{k} \sum_{t=1}^{k} \left(\text{orth}(V)_j^\top \text{orth}(V)_i^\top\right) = 0.$$

□

Lemma 5 (Reduced rank regression)**.** *Let $C, D \in \mathbb{R}^{n \times n}$, $A \in \mathbb{R}^{n \times k}$, and $B \in \mathbb{R}^{k \times n}$ be full rank matrices with $n \geq k$. Let $A^*, B^* = \min_{A,B} \|CAB - DC\|_F^2$. Let $u_1, \ldots, u_k$ and $v_1 \ldots, v_k$ be the top-k singular vectors of $C^{-1}DC$ according to Definition 12. Then* $\text{span}(A^*) = \text{span}(u_1, \ldots, u_k)$ *and* $\text{span}(B) = \text{span}(v_1, \ldots, v_k)$.

This is a reduced rank regression problem (Izenman 1975) or low-rank matrix approximation problem.

The unconstrained solution to the problem is given by $\hat{A}\hat{B} = C^{-1}DC$. From the Eckhart-Young theorem, we know that the optimal low-rank approximation to $C^{-1}DC$ is given by the top-k singular vectors. Therefore A and B span top-k left and right singular vectors respectively. For a more extensive proof, please refer to Izenman 1975.

A.6.2 Stochastic matrices

Lemma 6 (Spectrum of a resolvent matrix)**.** *Let A be an invertible matrix with unique real eigenvalues and $-1 \leq \lambda_{\min} \leq \lambda_{\max} \leq 1$. Let $\gamma \in (0, 1)$. The matrices A and $(I - \gamma A)^{-1}$ have the same eigenvectors and the ordering of the corresponding eigenvalues remains the same.*

Proof. Let e be an eigenvector of A and λ the corresponding eigenvalue. Then

$$(I - \gamma A)^{-1} e = \sum_{i=0}^{n} \gamma^n A^n e = \sum_{i=0}^{n} (\gamma\lambda)^n e = \frac{1}{1-\gamma\lambda} e.$$

As $(1-\gamma\lambda)^{-1}$ is a monotonic function for $-1 \leq \lambda \leq 1$ and $\gamma \in [0,1]$, the ordering of the eigenvalues remains the same. □

Lemma 7 (Basis equivalence of linear reward and value function)**.** *Let r be the vector representation of the reward function and V of the value function respectively for an MDP with fixed policy and transition matrix P^π. Let $U = \{u_1, \ldots, u_n\}$ be the set of eigenvectors of P^π, and let $U^r \subseteq U$ be a minimal set of eigenvectors so that $r \in \text{span}(U^r)$. Then $V \in \text{span}(U^r)$.*

Proof. Let $r = \sum_{i=1}^{k} \alpha_i u_i^r$ be the reward representation in the basis U^r. Then, by Lemma 6

$$V^\pi = (I - \gamma P^\pi)^{-1} r = \sum_{i=1}^{k} \frac{\alpha}{1-\lambda_i\gamma} u_i^r.$$

□

A.6.3 Ordinary differential equations

For describing the stability of ODEs, we use the notion of *asymptotic stability*, with the condition $\Re(\lambda_i) < 0$ for all eigenvalues λ_i of the Jacobian at a critical point.

Lemma 8 (Linear reparameterization of an autonomous ODE)**.** *Let $y' = f(y)$ be an autonomous ordinary differential equation. Let y^* be any critical point for which $f(y^*) = 0$. Let furthermore $x' = f(Ax)$ be a reparameterized autonomous ODE for any invertible matrix A. Then the x^* is a critical point with $f(Ax) = 0$ iff $x^* = A^{-1}y^*$. Furthermore, the eigenvalues of the Jacobian of y' at y^* are equal to the eigenvalues of the Jacobian of x' at $x^* = A^{-1}y^*$.*

Proof. The direction $x = A^{-1}y^* \Rightarrow f(Ax) = 0$ is clear by direct evaluation. We now focus on the direction $f(Ax) = 0 \Rightarrow x = A^{-1}y^*$. Assume that $f(Ax)$ is 0 and $x = A^{-1}y$ for a y which is not a point satisfying $f(y) = 0$. But then $0 = f(AA^{-1}y) = f(y) \neq 0$, a contradiction.

For stability, note that

$$\begin{aligned}
& & \frac{\mathrm{d}}{\mathrm{d}t}y &= f(y) \\
\implies & & \frac{\mathrm{d}}{\mathrm{d}t}x &= \frac{\mathrm{d}}{\mathrm{d}y}x\frac{\mathrm{d}}{\mathrm{d}t}y \\
& & &= A^{-\top}f(y) \\
\implies & \frac{\mathrm{d}}{\mathrm{d}x}\frac{\mathrm{d}}{\mathrm{d}t}x &= A^{-\top}\frac{\mathrm{d}}{\mathrm{d}x}f(Ax) \\
& & &= A^{-\top}\frac{\mathrm{d}}{\mathrm{d}y}f(y)\frac{\mathrm{d}}{\mathrm{d}x}Ax \\
& & &= A^{-\top}\frac{\mathrm{d}}{\mathrm{d}y}f(y)A^{\top}.
\end{aligned}$$

This shows that the Jacobian $\frac{\mathrm{d}}{\mathrm{d}x}f(Ax)|_{x=A^{-1}y_0}$ is similar to the Jacobian $\frac{\mathrm{d}}{\mathrm{d}y}f(y)|_{y=y_0}$, which means their spectra are identical. □

A.6.4 MDP representation and TD learning

Lemma 9 (Lemma 5 of Tang, Z. D. Guo, et al. 2022)**.** *Suppose P^π is real-diagonalizable, and write $u_1, \ldots, u_n$ for its eigenvectors. Then any orthonormal matrix Φ which has the same span as a set of k eigenvectors is a minimizer of L_{lat}.*

The next three statements are taken from Ghosh and Marc G Bellemare 2020. They address the TD loss wrt to the learned weights $\hat{V}$ and fixed Φ (compare Subsection 5.2.1). We changed the notation of the statements to fit our notation here, we have denoted the diagonal matrix of the state distribution as D and assume that $D = I$ in Assumption 1, while Ghosh and Marc G Bellemare 2020 uses Ξ. They use θ for the value function weights while we use $\hat{V}$. They also uses $\mathrm{Spec}(A)$ to denote the spectrum, the set of all eigenvalues of a matrix A.

The notion of stability used in Ghosh and Marc G Bellemare 2020 is that of convergence to the unique fixed point of the projected Bellman update of the linear ODE induced by the L_{TD} loss when fixing Φ. In the two-timescale scenario considered in this paper, this corresponds to the "inner" ODE over $\hat{V}$.

Lemma 10 (Proposition 3.1 of Ghosh and Marc G Bellemare 2020)**.** *TD(0) is stable if and only if the eigenvalues of the implied iteration matrix $A_\Phi = \Phi^\top D(I - \gamma P^\pi)\Phi$ have positive real components, that is*

$$Spec\,(A_\Phi) \subset \mathbb{C}_+ \coloneqq \{z : Re(z) > 0\}.$$

We say that a particular choice of representation Φ is stable for (P^π, γ, D) when A_Φ satisfies the above condition.

Lemma 11 (Proposition 3.2 of Ghosh and Marc G Bellemare 2020)**.** *An orthogonal representation Φ is stable if and only if the real part of the eigenvaluse of the induced transition matrix $\Pi P^\pi \Pi$ where $\Pi = \Phi\Phi^\top$ is bounded above, according to*

$$Spec\,(\Pi P^\pi \Pi) \subset \{z \in \mathbb{C} : Re(z) < \frac{1}{\gamma}\}.$$

In particular, Φ is stable if $\rho(\Pi P^\pi \Pi) < \frac{1}{\gamma}$.

Lemma 12 (Theorem 4.1 of Ghosh and Marc G Bellemare 2020)**.** *An orthogonal invariant representation Φ (meaning $span(P^\pi\Phi) \subseteq span(\Phi)$) satisfies*

$$Spec\,(\Pi P^\pi \Pi) \subseteq Spec(P^\pi) \cup \{0\}$$

and is therefore stable.

As a corollary of their proof we have that

Lemma 13 (Corollary of Ghosh and Marc G Bellemare 2020)**.** *Let Φ be an orthogonal (but not necessarily square) invariant representation of P^π. Then the spectral radius $\rho(\Phi^\top P^\pi \Phi) \leq 1$.*

The proof follows directly from Lemma 12 by the cyclicality of the spectrum.

The following two results are our own, although closely related results exist in the literature.

Lemma 14 (Lossless approximation of V)**.** *Let Φ be an orthonormal basis of an invariant subspace of P^π and let Assumption 4 hold. Let $V = (I - \gamma P^\pi)^{-1} r$ be the value function of P^π and r^π. Then*

$$\Phi(I - \gamma\Phi^\top P^\pi \Phi)^{-1}\Phi^\top r^\pi = V.$$

Proof. By Assumption 4 we can find a matrix A so that $r^\pi = \Phi A$ and $\Phi\Phi^\top r^\pi = r^\pi$, and by the invariant subspace assumption, we can find a matrix B so that $P^\pi\Phi = \Phi B$. Writing the inverted matrix as an infinite sum, which is valid as the spectrum of $\Phi^\top P^\pi \Phi$ is bounded by

1 following from Lemma 13 and Carl Neumann's theorem over power series, we obtain

$$\begin{aligned}
\Phi(I-\gamma\Phi^\top P^\pi\Phi)^{-1}\Phi r^\pi &= \Phi\sum_{n=0}^{\infty}\gamma^n(\Phi^\top P^\pi\Phi)^n\Phi r^\pi \\
&= \Phi\sum_{n=0}^{\infty}\gamma^n B^n\Phi^\top r^\pi && (P^\pi\Phi=\Phi B) \text{ and } (\Phi^\top\Phi=I) \\
&= \sum_{n=0}^{\infty}\gamma^n\Phi B^n\Phi^\top r^\pi \\
&= \sum_{n=0}^{\infty}\gamma^n P^{\pi n}\Phi\Phi^\top r^\pi && (\Phi B=P^\pi\Phi) \text{ iterated} \\
&= \sum_{n=0}^{\infty}\gamma^n P^{\pi n} r^\pi && (\Phi\Phi^\top r^\pi = r^\pi) \\
&= V
\end{aligned}$$

□

Lemma 15 (Critical points of L_{TD})**.** *Let Assumption 1, Assumption 3, and Assumption 4 hold. Assume $\Phi^* \in \mathbb{R}^{n\times k}$ is an orthonormal invariant representation for P^π in the sense that ${\Phi^*}^\top\Phi^* = I$ and span(Φ^*) = span$(P^\pi\Phi^*)$. Let furthermore $r^\pi \in$ span(Φ^*) and let V be the corresponding value function. Then Φ^* is a critical point of L_{TD} and for the corresponding weights $\hat{V}^*$ we have $\Phi^*\hat{V}^* = V$.*

Proof. By Lemma 10, Lemma 11, and Lemma 12 and the stated assumptions the weights $\hat{V}$ converge. Therefore, we can analyze the induced dynamical system with $\hat{V}*$.

Find $\hat{V}^*$ as the stationary point of $\nabla_{\hat{V}} L_{\text{TD}}$

$$\begin{aligned}
&\nabla_{\hat{V}}\left\|\Phi^*\hat{V} - \left[r^\pi+\gamma P\Phi^*\hat{V}\right]_{\text{sg}}\right\|_2^2\Big|_{\hat{V}=\hat{V}^*} = {\Phi^*}^\top\left(\Phi^*\hat{V}^* - \left[r^\pi+\gamma P\Phi^*\hat{V}^*\right]\right) = 0 \\
\Leftrightarrow\quad & {\Phi^*}^\top\Phi^*\hat{V}^* - {\Phi^*}^T r^\pi - \gamma{\Phi^*}^\top\Phi^* B\hat{V}^* = ({\Phi^*}^\top\Phi^* - \gamma{\Phi^*}^\top P^\pi\Phi^*)\hat{V}^* - {\Phi^*}^\top r^\pi = 0 \\
\Leftrightarrow\quad & \hat{V}^* = ({\Phi^*}^\top\Phi^* - \gamma{\Phi^*}^\top P^\pi\Phi^*)^{-1}{\Phi^*}^\top r^\pi = (I-\gamma{\Phi^*}^\top P^\pi\Phi^*)^{-1}{\Phi^*}^\top r^\pi
\end{aligned}$$

The invertibility of $(I-\gamma\Phi^\top P^\pi\Phi) = \Phi^T D(I-\gamma P^\pi)\Phi = A_\Phi$ is guaranteed as all eigenvectors are nonzero.

We now show that Φ^* is a stationary point by showing that

$$\nabla_\Phi L_{\text{TD}}|_{\Phi=\Phi^*} = 0.$$

We note that as Φ^* spans an invariant subspace of P^π there exists an invertible matrix B so that $P^\pi\Phi^*\Phi^*B$. Therefore $(I-\gamma{\Phi^*}^\top P^\pi\Phi^*) = (I-\gamma B)$.

$$
\begin{aligned}
\nabla_\Phi L_{\mathrm{TD}}|_{\Phi=\Phi^*} &= \nabla_\Phi \left\| \Phi\hat{V}^* - \left[r^\pi + \gamma P \Phi \hat{V}^*\right]_{\mathrm{sg}} \right\|_2^2 \Big|_{\Phi=\Phi^*} \\
&= \left(\Phi\hat{V}^* - r^\pi - \gamma P \Phi \hat{V}^*\right)(\hat{V}^*)^\top \Big|_{\Phi=\Phi^*} \\
&= \left(\Phi^*(I-\gamma B)\hat{V}^* - r^\pi\right)(\hat{V}^*)^\top \qquad \text{(substitute first occurrence of) } \hat{V}^* \\
&= \left(\Phi^*\Phi^{*\top} r^\pi - r^\pi\right)(\hat{V}^*)^\top \\
&= \underbrace{(r^\pi - r^\pi)}_{=0}(\hat{V}^*)^\top = 0
\end{aligned}
$$

The final line is due to the fact that the columns of Φ^* are orthonormal, which means that $\Phi^*\Phi^{*\top}$ is an orthogonal projection onto the span of Φ^*. To verify, note that

$$
\left(\Phi^*\Phi^{*\top}\right)^2 = \left(\Phi^* \underbrace{\Phi^{*\top}\Phi^*}_{=I} \Phi^{*\top}\right) = \left(\Phi^*\Phi^{*\top}\right).
$$

Furthermore, by Assumption 4, $r^\pi \in \mathrm{span}(\Phi^*)$, which means $\Phi^*\Phi^{*\top} r^\pi = r^\pi$ for an orthogonal projection $\Phi^*\Phi^{*\top}$. Moreover, by Lemma 14, $\Phi^*\hat{V}^* = V$, which concludes the proof. □

This proof closely follows related statements by Ghosh and Marc G Bellemare 2020, Tang, Z. D. Guo, et al. 2022, and Le Lan, Tu, Oberman, et al. 2022. We repeated the argument here for easier legibility with all assumptions necessary for our work.

A.7 Proofs of main results

Proofs for Section 4

Proposition 2 (Stationary points of latent self-prediction)**.** *Assume Assumption 1 and Assumption 2 hold. Furthermore, suppose P^π is real diagonalizable. If the columns of Φ_t span an invariant subspace of P^π, Φ_t is a stationary point of the dynamical system. Furthermore, if P^π is real-diagonalizable with positive eigenvalues, all invariant subspaces not spanned by the top-k eigenvectors sorted by eigenvalue are asymptotically unstable for gradient descent.*

Proof. At any stationary point the gradient $\frac{d}{dt}\Phi_t$ must be equal to 0, which from Section 5.2.1 means that we must have $\left(\Phi_t\left(\Phi_t^\top\Phi\right)^{-1}\Phi_t^\top - I\right) P^\pi \Phi_t \Phi_t^\top P^{\pi\top} \Phi_t \left(\Phi_t^\top\Phi\right)^{-\top} = 0$.

Assume that the column vectors of Φ^* spans an invariant subspace of P^π. This implies that

there exists a full rank matrix A so that $P^\pi \Phi^* = \Phi^* A$. Then

$$\left(\Phi^* \left(\Phi^{*\top}\Phi^*\right)^{-1} \Phi^{*\top} - I\right) P^\pi \Phi^* F^* = \left(\Phi^* \underbrace{\left(\Phi^{*\top}\Phi^*\right)^{-1} \Phi^{*\top}\Phi^*}_{=I} - \Phi^*\right) A F^* = 0.$$

This proves the first part of the proposition.

There are additional critical points of the differential equation, as discussed by Tang, Z. D. Guo, et al. 2022. In the analysis of stability, we first show the case of critical points corresponding to the claim in the proposition. We then briefly discuss other cases after the proof.

Case 1: Φ_t spans an invariant subspace of P^π Invariant subspaces correspond to subspaces spanned by right eigenvectors of P^π.

We write P for P^π to reduce notational clutter. Let $e_1, \ldots, e_k$ be the eigenvectors corresponding to the k largest eigenvalues of P. Let Φ^* correspond to any set of k eigenvectors of P. Then

$$\frac{\mathrm{d}}{\mathrm{d}t}\Phi^* = -\left(\Phi^* F^* - P\Phi^*\right) F^{*\top} = 0.$$

To show that all non top-k eigenspaces are asymptotically unstable critical points of the differential equation defined by the gradient flow of Φ. To show this, we aim to show that there exists an eigenvector of the Jacobian with an eigenvalue larger than 0. For this, we construct the directional derivative at the critical point. The directional derivative is the Jacobian vector product, which allows us to circumvent the need to work with higher order tensor derivatives. We then proceed to show that there exists a direction which corresponds to the eigenvector of the Jacobian with positive eigenvalue. This concludes the proof. This technique closely follows the one used by Le Lan, Tu, Rowland, et al. 2023.

Assume $\mathrm{span}\{\Phi^*\} \neq \mathrm{span}\{e_1, \ldots, e_k\}$. This implies that there exists at least one eigenvector $e_j \in \{e_1, \ldots, e_k\}$ and $e_j \notin \mathrm{span}\{\Phi^*\}$, with corresponding eigenvalues λ_j.

Let D_Δ be the directional derivative of $\frac{\mathrm{d}}{\mathrm{d}t}\Phi|_{\Phi=\Phi^*}$ in the direction Δ. We construct the directional derivative using the product rule (terms colored for ease of reading),

$$\begin{aligned} D_\Delta \frac{\mathrm{d}}{\mathrm{d}t}\Phi|_{\Phi=\Phi^*} &= -D_\Delta\left(\left(\Phi F^* - P\Phi\right) F^{*\top}\right)|_{\Phi=\Phi^*} \\ &= -D_\Delta\left(\Phi F^* - P\Phi\right)|_{\Phi=\Phi^*} F^{*\top} - \left(\Phi^* F^* - P\Phi^*\right) D_\Delta F^*|_{\Phi=\Phi^*}{}^\top. \end{aligned}$$

For the directional derivative, we only consider directions that are orthogonal to Φ^*, so

${\Phi^*}^\top \Delta = 0$. Then $\underbrace{P\Phi^* = \Phi^* A}_{\text{subspace condition}} \implies \Delta^\top P \Phi^* = 0$. For the derivative with regard to F^*, we have

$$\begin{aligned} D_\Delta F^*|_{\Phi=\Phi^*} =& D_\Delta \left(\Phi^\top \Phi\right)^{-1} \Phi^\top P \Phi \\ =& \left(D_\Delta \left(\Phi^\top \Phi\right)^{-1}\right) {\Phi^*}^\top P \Phi^* + \left({\Phi^*}^\top \Phi^*\right)^{-1} \left(D_\Delta \Phi^\top P \Phi\right) \\ =& -\underbrace{(\Delta^\top \Phi^* + {\Phi^*}^\top \Delta)}_{=0} \left({\Phi^*}^\top \Phi^*\right)^{-2} {\Phi^*}^\top P \Phi^* + \left({\Phi^*}^\top \Phi^*\right)^{-1} (\underbrace{\Delta^\top P \Phi^*}_{=0} + {\Phi^*}^\top P \Delta) \\ =& \left({\Phi^*}^\top \Phi^*\right)^{-1} \left({\Phi^*}^\top P \Delta\right). \end{aligned}$$

Therefore, the first term in the second line is dropped, as well as the first term of the final summand.

Note that $F^* = \left({\Phi^*}^\top \Phi^*\right)^{-1} {\Phi^*}^\top P^\pi \Phi^* = \underbrace{\left({\Phi^*}^\top \Phi^*\right)^{-1} {\Phi^*}^\top \Phi^*}_{=I} \operatorname{diag}(\Lambda_i) = \operatorname{diag}(\Lambda_i)$, where Λ_i is the set of eigenvalues corresponding to the eigenvectors in Φ^* and $\operatorname{diag}(\Lambda_i)$ is the diagonal matrix of eigenvalues corresponding to those eigenvectors spanned by Φ^*.

This allows us to compute the remaining derivative,

$$\begin{aligned} & D_\Delta \left(\Phi F^* - P\Phi\right)|_{\Phi=\Phi^*} \operatorname{diag}(\Lambda_i) \\ =& \left(\Delta \operatorname{diag}(\Lambda_i) + \Phi^* D_\Delta F^* - P\Delta\right) \operatorname{diag}(\Lambda_i) \\ =& \left(\Delta \operatorname{diag}(\Lambda_i) + \Phi^* \left({\Phi^*}^\top \Phi^*\right)^{-1} {\Phi^*}^\top P\Delta - P\Delta\right) \operatorname{diag}(\Lambda_i), \end{aligned}$$

where we use the fact that $D_\Delta P\Phi^* = PD_\Delta \Phi^* = P\Delta$.

Finally, as $\Phi^* F^* = \Phi^* \operatorname{diag}(\Lambda_i) = P^\pi \Phi^*$, we obtain

$$D_\Delta \frac{\mathrm{d}}{\mathrm{d}t}\Phi|_{\Phi=\Phi^*} = -\left(\Delta \operatorname{diag}(\Lambda_i) + \Phi^* \left({\Phi^*}^\top \Phi^*\right)^{-1} {\Phi^*}^\top P\Delta - P\Delta\right) \operatorname{diag}(\Lambda_i) - \underbrace{(\Phi^* F^* - P\Phi^*)}_{=0 \text{ as shown}} \left({\Phi^*}^\top P\Delta\right)^\top.$$

By the definition of the directional derivative as the Jacobian-vector product, we can now assert

$$\left(\frac{\mathrm{d}}{\mathrm{d}\Phi}\frac{\mathrm{d}}{\mathrm{d}t}\Phi|_{\Phi=\Phi^*}\right)\Delta = -\left(\Delta \operatorname{diag}(\Lambda_i) + \left(\Phi^* \left({\Phi^*}^\top \Phi^*\right)^{-1} {\Phi^*}^\top - I\right) P\Delta\right) \operatorname{diag}(\Lambda_i).$$

What remains to be shown is that there exist a direction which corresponds to a positive eigenvalue of the Jacobian of the dynamics.

Choose $\Delta = v_j u^\top$. Let v_j be an eigenvector not in the span of Φ^* but in the top-k

eigenvectors. Let λ_j be the corresponding eigenvalue. By our assumption before, there exist at least one eigenvalue $\lambda_i \in \Lambda_i$ so that $\lambda_j > \lambda_i$.

Note that $\left(\Phi^* \left(\Phi^{*\top}\Phi^*\right)^{-1} \Phi^{*\top} - I\right) P v_j = \Big(\Phi \left(\Phi^{*\top}\Phi^*\right)^{-1} \underbrace{\Phi^{*\top} v_j}_{0 \text{ by construction}} - I v_j\Big)\lambda_j = -\lambda_j v_j$

and therefore $\left(\Phi^* \left(\Phi^{*\top}\Phi^*\right)^{-1} \Phi^{*\top} - I\right) P\Delta = -\Delta\lambda_j I$.

To simplify notation, we will write Λ_i for $\text{diag}(\Lambda_i)$ from now on as there is no risk of confusion.

$$
\begin{aligned}
-\left(\Delta\Lambda_i + \left(\Phi^* \left(\Phi^{*\top}\Phi^*\right)^{-1} \Phi^{*\top} - I\right) P\Delta\right)\Lambda_i =& -\Delta\left(\Lambda_i - \lambda_j I\right)\Lambda_i \\
=& -\Delta\left(\Lambda_i^2 - \lambda_j\Lambda_i\right) \\
=& \Delta\left(\lambda_j\Lambda_i - \Lambda_i^2\right).
\end{aligned}
$$

We can now choose u so that it is any eigenvector of $\left(\lambda_j\Lambda_i - \Lambda_i^2\right)$. As this is a diagonal matrix, it is easy to see that if $\lambda_j > \lambda_i$ for any λ_i, the matrix will have a positive eigenvalue, meaning there exists a direction in which the critical point is unstable.

□

Case 2: Non-invariant subspace cases: Not all critical points lie in invariant subspaces. One such an alternative critical point is the case of $\Phi^\top P\Phi = 0$, and more generally, for each set of column vectors ϕ_i in Φ, they needs to either be mapped to an invariant or an orthogonal subspace by P^π to be stable. In the orthogonal case, the Jacobian at the critical point becomes 0, meaning no conclusion about stability can be drawn from this analysis.

We leave further analysis of non invariant subspace critical points open for future work. We do however conjecture that the non invariant subspace critical points are also saddle-points or unstable solutions of the ODE, following the experimental analysis by Tang, Z. D. Guo, et al. 2022.

Negative eigenvalues: In case the matrix has negative eigenvalues, the stability conditions in the final step of the proof change. The matrix $\lambda_j\Lambda_i - \Lambda_i^2$ will not have negative eigenvalues if λ_i is negative but λ_j is positive. The ranking of stable points follows this slightly unintuitive ordering: all negative eigenvalues sorted by absolute value followed by all positive eigenvalues sorted by absolute value.

Proposition 3 (Stationary points of reconstruction)**.** *Assume Assumption 1 and Assumption 2 hold. Write* $(u_1, \ldots, u_n)$, $(v_1, \ldots, v_n)$ *for the left and right singular vectors of* P^π

sorted in descending order by singular value. Any stationary point (Φ^*, F^*, Ψ^*) *of* L_{rec} *under the two timescale scenario satisfies* $\text{span}\,(\Phi^*) = \text{span}\,(\{u_1, \ldots, u_k\})$, $\text{span}\,(\Psi^{*\top}) = \text{span}\,(\{v_1, \ldots, v_k\})$.

Proof. We first show that under the two timescale scenario, F is stationary and therefore does not change the span of the critical points.

Due to the assumption of the two-timescale scenario, we compute Ψ^* by solving the linear regression problem,

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\Psi^*}\|\Phi F\Psi^* - P\|_F^2 &= F^\top\Phi^\top(\Phi F\Psi^* - P) \\
0 &= F^\top\Phi^\top(\Phi F\Psi^* - P) \\
\Leftrightarrow \qquad B^* &= \left(F^\top\Phi^\top\Phi F\right)^{-1} F^\top\Phi^\top P = F^{-1}\left(\Phi^\top\Phi\right)^{-1}\Phi^\top P.
\end{aligned}$$

Plugging this solution back into the original equation,

$$\begin{aligned}
\|\Phi F\Psi^* - P\|_F^2 &= \|\Phi F F^{-1}\left(\Phi^\top\Phi\right)^{-1}\Phi^\top P - P\|_F^2 \\
&= \|\Phi\left(\Phi^\top\Phi\right)^{-1}\Phi^\top P - P\|_F^2,
\end{aligned}$$

we notice that F cancels. Therefore, the optimality conditions for A follow from the Eckart-Young theorem, as presented in Lemma 5 □

Proposition 4. *Assume Assumption 1, Assumption 2, and Assumption 3 hold. Let* $\{\Phi^*_{\text{lat/td}}\}$ *be the set of critical points of* L_{lat} *or* L_{td} *respectively. Then* $\mathcal{O}^{-1}\Phi^*_{\text{lat/td}}$ *are stationary points for the reparameterized losses* $L^{\mathcal{O}}_{\text{lat}}$ *and* $L^{\mathcal{O}}_{\text{td}}$.[1] *Furthermore, if* $\Phi^*_{\text{lat/td}}$ *is an asymptotically stable point of* $L_{\text{lat/td}}$ *that has a Jacobian with all negative eigenvalues,* $\mathcal{O}^{-1}\Phi^*_{\text{lat/td}}$ *is an asymptotically stable point of* $L^{\mathcal{O}}_{\text{lat/td}}$.

Proof. We first write out all losses with the observation matrix $\mathcal{O}$. The reward function is assumed to not change under the introduction of $\mathcal{O}$, therefore we do not multiply $\mathcal{O}$ to $x^\top r$.

$$\begin{aligned}
L_{\text{rec}}(\Phi, F, \Psi) =& \mathbb{E}_{x\sim\mathcal{D}}\left[\left\|x^\top\mathcal{O}\Phi F\Psi - x^\top P^\pi\mathcal{O}\right\|_2^2\right] = \|\mathcal{O}\Phi F\Psi - P^\pi\mathcal{O}\|_2^2, \\
L_{\text{lat}}(\Phi, F) =& \mathbb{E}_{x\sim\mathcal{D}}\left[\left\|x^\top\mathcal{O}\Phi F - \left[x^\top P^\pi\mathcal{O}\Phi\right]_{\text{sg}}\right\|_2^2\right] = \left\|\mathcal{O}\Phi F - \left[P^\pi\mathcal{O}\Phi\right]_{\text{sg}}\right\|_2^2 \\
L_{\text{td}}(\Phi, F) =& \mathbb{E}_{x\sim\mathcal{D}}\left[\left\|x^\top\mathcal{O}\Phi\hat{V} - \left[x^\top\left(r + \gamma P^\pi\mathcal{O}\Phi\hat{V}\right)\right]_{\text{sg}}\right\|_2^2\right] = \left\|\mathcal{O}\Phi\hat{V} - \left[\left(r + \gamma P^\pi\mathcal{O}\Phi\hat{V}\right)\right]_{\text{sg}}\right\|_2^2.
\end{aligned}$$

[1] Due to space constraints, we present the full equations in the proof.

Note that in the cases of L_{lat} and L_{TD}, all occurrences of $\mathcal{O}$ are multiplied by Φ. Therefore the corresponding gradient flows are reparameterizations as defined in Lemma 8, and the proof follows directly. □

Proposition 5. *Assume Assumption 1, Assumption 2, and Assumption 3 hold. Let* $(u_1, \ldots, u_n)$, $(v_1, \ldots, v_n)$ *be the left and right singular vectors of* $\mathcal{O}^{-1}P^\pi\mathcal{O}$. *Any stationary point* (Φ^*, F^*, Ψ^*) *of* $L^{\mathcal{O}}_{rec}$ *satisfies* $\text{span}\,(\Phi^*) = \text{span}\,(\{u_1, \ldots, u_k\})$, $\text{span}\,(\Psi^{*\top}) = \text{span}\,(\{v_1, \ldots, v_k\})$.

Proof. As before, note that $L^{\mathcal{O}}_{\text{rec}}$ is of the form $\|\mathcal{O}AXB - P\mathcal{O}\|_F^2$, with $A \in \mathbb{R}^{n\times k}$, $X \in \mathbb{R}^{k\times k}$, and $B \in \mathbb{R}^{k\times n}$.

Due to the assumption of the two-timescale scenario, we compute Ψ^* by solving the linear regression problem,

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}\psi}\|\mathcal{O}\Phi F\Psi - P\mathcal{O}\|_F^2 &= F^\top\Phi^\top\mathcal{O}^\top(\mathcal{O}\Phi F\Psi - P) \\
0 &= F^\top\Phi^\top\mathcal{O}^\top(\mathcal{O}\Phi F\Psi^* - P) = 0 \\
\Leftrightarrow \qquad \Psi^* &= \left(F^\top\Phi^\top\mathcal{O}^\top\mathcal{O}\Phi F\right)^{-1} F^\top\Phi^\top\mathcal{O}^\top P.
\end{aligned}$$

Substituting into $L^{\mathcal{O}}_{\text{rec}}$, we obtain

$$\begin{aligned}
\|\mathcal{O}\Phi F\Psi - P\mathcal{O}\|_F^2 &= \|\mathcal{O}\Phi F\left(F^\top\Phi^\top\mathcal{O}^\top\mathcal{O}\Phi F\right)^{-1} F^\top\Phi^\top\mathcal{O}^\top P - P\mathcal{O}\|_F^2 \\
&= \|\mathcal{O}\Phi\left(\Phi^\top\mathcal{O}^\top\mathcal{O}\Phi\right)^{-1}\Phi^\top\mathcal{O}^\top P - P\mathcal{O}\|_F^2,
\end{aligned}$$

which again implies that F is stationary.

We note that $\|\mathcal{O}\Phi\Psi - P\mathcal{O}\|_F^2$ is the reduced rank regression problem solved in Lemma 5 which solution is given by the top-k left and right singular vectors of $\mathcal{O}^{-1}P\mathcal{O}$. □

Proofs for Section 5

Proposition 6. *Suppose that* P^π *is real diagonalizable, and that Assumption 1, Assumption 3, and Assumption 4 hold. There exists a non-trivial critical point* Φ^* *of the two-timescale TD loss* L_{td} *such that* $\text{span}(r^\pi) \subseteq \text{span}(\Phi^*)$. *Furthermore,* Φ^* *is a critical point of the two-timescale joint loss* L_{td+lat}. *Therefore combining* L_{TD} *and* L_{Lat} *does not exclude the existence of a stationary point with* 0 *value function approximation error.*

Proof. By Assumption 4 there exists a set of k vectors $\phi_1, \ldots, \phi_k$ such that $r^\pi \in \text{span}(\phi_1, \ldots, \phi_k)$ and $\phi_1, \ldots, \phi_k$ span an invariant subspace of P^π. We can choose this set of vectors to orthonormal, e.g. by applying the Gram Schmidt procedure to the eigenvectors $(w_{i1}, \ldots w_{im})$.

By Lemma 9 the matrix $\Phi \in \mathbb{R}^{n\times k}$ whose columns are $\phi_1, \ldots, \phi_k$ is a critical point for L_{lat} and spans an invariant subspace of P^π. In addition, by Lemma 7, Φ forms a complete basis for the value function V^π. By Lemma 15, Φ is also a critical point of L_{TD}, with 0 value function approximation error. As Φ is a critical point for both L_{TD} and L_{Lat}, it is a critical point of $L_{\text{TD}} + L_{\text{Lat}}$.

□

Proposition 7. *Let Assumption 1, Assumption 2, and Assumption 4 hold. If the reward spanning eigenvectors do not lie within the span of the top-k singular vectors,* $\text{span}\,(w_{i_1}, \ldots, w_{i_m}) \not\subseteq \text{span}\,(u_1, \ldots, u_k)$*, the critical points of the two-timescale joint loss* L_{td+rec} *are guaranteed to not be minimizers of the value function approximation error.*

Proof. Following from Assumption 4 and Lemma 7, we have that any critical point Φ^* of L_{td} with perfect value function approximation fulfils $\text{span}(r^\pi) \subseteq \text{span}(\Phi^*)$, as without this condition, Φ would not have all necessary basis vectors to represent V^π. Under our low-dimensionality assumption $r^\pi \in \text{span}(w_{i_1}, \ldots, w_{i_m})$, this condition implies that $\text{span}(w_{i_1}, \ldots, w_{i_m}) \subseteq \text{span}(\Phi^*)$. From Proposition 9 we know that any critical point Φ^* of L_{rec} satisfies $\text{span}\,(\Phi^*) = \text{span}\,(\{u_1, \ldots, u_k\})$, where $u_1, \ldots, u_k$ are the top k left singular vectors of P^π. Under the assumption that $\text{span}(w_{i_1}, \ldots, w_{i_m}) \not\subseteq \text{span}(u_1, \ldots, u_k)$ these conditions cannot happen simultaneously, and hence no critical point of the joint loss achieves perfect value function reconstruction. □

Proposition 5 (Suboptimality of top k eigenspaces with distractions)**.** *Assume an MDP with distraction composed of two independent processes* $\mathcal{M}$ *and* $\mathcal{N}$ *according to Definition 11. Let* $v_1, \ldots, v_n$ *and* $u_1, \ldots, u_m$ *be the eigenvectors of* $\mathcal{M}$ *and* $\mathcal{N}$ *respectively, with associated real eigenvalues* λ_j *and* μ_i *ordered. Assume* $\forall i < k : \mu_i > \lambda_2$ *and* $r_{\mathcal{N}} = 0$*. Let* U_k *be an orthonormal basis for the top-k eigenspace of* $\mathcal{M}\otimes\mathcal{N}$*. Then* $\text{span}(U_k) = \text{span}(\{\mathbf{1}\otimes v_i | \forall i \leq k\})$ *and* $\text{proj}_{U_k}(r_{\mathcal{M}} \otimes \mathbf{1}) = (\sum_{i=0}^{m} r_i/n)\mathbf{1}_{n\cdot m}$*.*

Proof. We first note that $\text{span}(V_k) = \text{span}(\{v_i \otimes u_1 | \forall i \leq k\})$ follows directly from Lemma 3 and Lemma 4. The eigenvector $u_1 = \mathbf{1}$ as M is a stochastic matrix.

As V_k is an orthogonal basis, write the projection operation as

$$\begin{aligned} V_k V_k^\top (r_{\mathcal{M}} \otimes \mathbf{1}) &= \frac{1}{m} (\mathbf{1} \otimes \text{orth}(V)) \left(\mathbf{1} \otimes \text{orth}(V)^\top\right) (r_{\mathcal{M}} \otimes \mathbf{1}) \\ &= \frac{1}{m} \left((\mathbf{1} \otimes \mathbf{1}) \otimes (\text{orth}(V)\, \text{orth}(V)^\top)\right) (r_{\mathcal{M}} \otimes \mathbf{1}) \\ &= \frac{1}{m} (\mathbf{1} \otimes \mathbf{1}) r_{\mathcal{M}} \otimes (\text{orth}(V)\, \text{orth}(V)^\top) \mathbf{1} \\ &= \sum_{i=1}^{m} \frac{r_i}{m} \mathbf{1}. \end{aligned}$$

□

Bibliography

Abachi, Romina, Mohammad Ghavamzadeh, and Amir-massoud Farahmand (2020). "Policy-Aware Model Learning for Policy Gradient Methods". In: *ArXiv* abs/2003.00030.

Abachi, Romina, Claas A Voelcker, et al. (2022). "VIPer: Iterative Value-Aware Model Learning on the Value Improvement Path". In: *Decision Awareness in Reinforcement Learning Workshop at ICML 2022*.

Abbas, Zaheer, Samuel Sokota, et al. (2020). "Selective Dyna-Style Planning Under Limited Model Capacity". In: *International Conference on Machine Learning*.

Abbas, Zaheer, Rosie Zhao, et al. (2023). *Loss of Plasticity in Continual Deep Reinforcement Learning*. arXiv: 2303.07507 [cs.LG].

Abel, David (2020). "A Theory of Abstraction in Reinforcement Learning". In: *PhD Thesis*.

Agarwal, Rishabh et al. (2021). "Deep Reinforcement Learning at the Edge of the Statistical Precipice". In: *Advances in Neural Information Processing Systems*.

Amos, Brandon et al. (2021). "On the model-based stochastic value gradient for continuous reinforcement learning". In: *Learning for Dynamics and Control*. PMLR.

Anderson, Charles (1992). "Q-Learning with Hidden-Unit Restarting". In: *Advances in Neural Information Processing Systems*.

Anschel, Oron, Nir Baram, and Nahum Shimkin (2017). "Averaged-DQN: Variance Reduction and Stabilization for Deep Reinforcement Learning". In: *International Conference on Machine Learning*.

Antonoglou, Ioannis et al. (2022). "Planning in Stochastic Environments with a Learned Model". In: *International Conference on Learning Representations*.

Asadi, Kavosh et al. (2018). "Equivalence Between Wasserstein and Value-Aware Loss for Model-based Reinforcement Learning". In: *ArXiv* abs/1806.01265.

Atkeson, Christopher G. and Stefan Schaal (1997). "Robot Learning From Demonstration". In: *International Conference on Machine Learning*.

Ayoub, Alex et al. (2020). "Model-based reinforcement learning with value-targeted regression". In: *International Conference on Machine Learning*.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton (2016). *Layer Normalization*. arXiv: 1607.06450 [stat.ML].

Baird, Leemon (1995). "Residual algorithms: Reinforcement learning with function approximation". In: *Machine learning proceedings 1995*. Elsevier, pp. 30–37.

Baldi, Pierre and Kurt Hornik (1989). "Neural networks and principal component analysis: Learning from examples without local minima". In: *Neural networks* 2.1, pp. 53–58.

Ball, Philip J. et al. (2023). "Efficient online reinforcement learning with offline data". In: *International Conference on Machine Learning*.

Bao, Xuchan et al. (2020). "Regularized linear autoencoders recover the principal components, eventually". In: *Advances in Neural Information Processing Systems*.

Barreto, André et al. (2017). "Successor features for transfer in reinforcement learning". In: *Advances in neural information processing systems*.

Behzadian, Bahram, Soheil Gharatappeh, and Marek Petrik (2019). "Fast feature selection for linear value function approximation". In: *International Conference on Automated Planning and Scheduling*.

Bellemare, Marc G et al. (2019). "A geometric perspective on optimal representations for reinforcement learning". In: *Advances in neural information processing systems*.

Bellemare, Marc G., Will Dabney, and Rémi Munos (2017). "A Distributional Perspective on Reinforcement Learning". In: *International Conference on Machine Learning*.

Bellman, Richard Ernest (1953). *An Introduction to the Theory of Dynamic Programming*. Santa Monica, CA: RAND Corporation.

Bengio, Yoshua, Aaron Courville, and Pascal Vincent (Aug. 2013). "Representation Learning: A Review and New Perspectives". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 35.8, pp. 1798–1828.

Bertsekas, Dimitri P. and Steven E. Shreve (1978). *Stochastic Optimal Control: The Discrete-Time Case*. Academic Press.

Bhatt, Aditya et al. (2024). "Cross$Q$: Batch Normalization in Deep Reinforcement Learning for Greater Sample Efficiency and Simplicity". In: *The Twelfth International Conference on Learning Representations*.

Bishop, Christopher M. (2006). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag. ISBN: 0387310738.

Bjorck, Johan, Carla P Gomes, and Kilian Q Weinberger (2022). "Is High Variance Unavoidable in RL? A Case Study in Continuous Control". In: *International Conference on Learning Representations*.

Borsa, Diana et al. (2019). "Universal Successor Features Approximators". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=S1VWjiRcKX.

Boutilier, Craig, Richard Dearden, and Moisés Goldszmidt (2000). "Stochastic dynamic programming with factored representations". In: *Artificial intelligence*.

Brockman, Greg et al. (2016). "OpenAI gym". In: *ArXiv* abs/1606.01540.

Buckman, Jacob et al. (2018). "Sample-efficient reinforcement learning with stochastic ensemble value expansion". In: *Advances in neural information processing systems*.

Chen, Xinyue, Che Wang, Zijian Zhou, and Keith W Ross (2020). "Randomized Ensembled Double Q-Learning: Learning Fast Without a Model". In: *International Conference on Learning Representations*.

— (2021). "Randomized Ensembled Double Q-Learning: Learning Fast Without a Model". In: *International Conference on Learning Representations*.

Chua, Kurtland et al. (2018). "Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models". In: *Advances in Neural Information Processing Systems*.

Cui, Brandon, Yinlam Chow, and Mohammad Ghavamzadeh (2021). "Control-Aware Representations for Model-based Reinforcement Learning". In: *International Conference on Learning Representations*.

D'Oro, Pierluca, Alberto Maria Metelli, et al. (2020). "Gradient-aware model-based policy search". In: *AAAI Conference on Artificial Intelligence*.

D'Oro, Pierluca, Max Schwarzer, et al. (2023). "Sample-Efficient Reinforcement Learning by Breaking the Replay Ratio Barrier". In: *International Conference on Learning Representations*.

Deisenroth, Marc and Carl E Rasmussen (2011). "PILCO: A model-based and data-efficient approach to policy search". In: *International Conference on Machine Learning*.

Ernst, Damien, Pierre Geurts, and Louis Wehenkel (2005). "Tree-based batch mode reinforcement learning". In: *Journal of Machine Learning Research* 6.

Eysenbach, Benjamin et al. (2022). "Mismatched No More: Joint Model-Policy Optimization for Model-Based RL". In: *Advances in Neural Information Processing Systems*.

Farahmand, Amir-massoud (2018). "Iterative Value-Aware Model Learning". In: *Advances in Neural Information Processing Systems*.

— (2021). *Lecture Notes on Reinforcement Learning*.

Farahmand, Amir-massoud, André Barreto, and Daniel Nikovski (2017). "Value-Aware Loss Function for Model-based Reinforcement Learning". In: *International Conference on Artificial Intelligence and Statistics*.

Farahmand, Amir-massoud, Csaba Szepesvári, and Rémi Munos (2010). "Error propagation for approximate policy and value iteration". In: *Advances in Neural Information Processing Systems*.

Farebrother, Jesse, Joshua Greaves, et al. (2023). "Proto-Value Networks: Scaling Representation Learning with Auxiliary Tasks". In: *International Conference on Learning Representations*.

Farebrother, Jesse, Marlos C. Machado, and Michael Bowling (2018). "Generalization and Regularization in DQN". In: *CoRR* abs/1810.00123. arXiv: 1810.00123.

Farebrother, Jesse, Jordi Orbay, et al. (2024). "Stop Regressing: Training Value Functions via Classification for Scalable Deep RL". In: *International Conference on Machine Learning*.

Fedus, William et al. (2020). "Revisiting fundamentals of experience replay". In: *International Conference on Machine Learning*.

Feinberg, Vladimir et al. (2018). "Model-based value estimation for efficient model-free reinforcement learning". In: *arXiv preprint arXiv:1803.00101*.

Ferns, Norm, Prakash Panangaden, and Doina Precup (2004). "Metrics for Finite Markov Decision Processes." In: *Uncertainty in AI*.

— (2011). "Bisimulation metrics for continuous Markov decision processes". In: *SIAM Journal on Computing* 40.6.

Fox, Roy, Ari Pakman, and Naftali Tishby (2016). "Taming the noise in reinforcement learning via soft updates". In: *Conference on Uncertainty in Artificial Intelligence*.

Fujimoto, Scott, Wei-Di Chang, et al. (2024). "For sale: State-action representation learning for deep reinforcement learning". In: *Advances in Neural Information Processing Systems* 36.

Fujimoto, Scott and Shixiang Gu (2021). "A Minimalist Approach to Offline Reinforcement Learning". In: *Advances in Neural Information Processing Systems*.

Fujimoto, Scott, Herke van Hoof, and David Meger (2018). "Addressing Function Approximation Error in Actor-Critic Methods". In: *International Conference on Machine Learning*.

Fujimoto, Scott, David Meger, and Doina Precup (2019). "Off-Policy Deep Reinforcement Learning without Exploration". In: *International Conference on Machine Learning*, pp. 2052–2062.

Gelada, Carles et al. (2019). "Deepmdp: Learning continuous latent space models for representation learning". In: *International Conference on Machine Learning*.

Ghosh, Dibya and Marc G Bellemare (2020). "Representations for stable off-policy reinforcement learning". In: *International Conference on Machine Learning*.

Ghugare, Raj et al. (2023). "Simplifying Model-based RL: Learning Representations, Latent-space Models, and Policies with One Objective". In: *International Conference on Learning Representations*.

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio (2011). "Deep Sparse Rectifier Neural Networks". In: *International Conference on Artificial Intelligence and Statistics*.

Goldberg, Yoav and Omer Levy (2014). *word2vec Explained: deriving Mikolov et al.'s negative-sampling word-embedding method*. arXiv: 1402.3722 [cs.CL].

Goodfellow, Ian et al. (2016). *Deep learning*. Vol. 1. MIT Press.

Gordon, Geoffrey J (1995). "Stable Function Approximation in Dynamic Programming". In: *International Conference on Machine Learning*.

Grill, Jean-Bastien et al. (2020). "Bootstrap your own latent-a new approach to self-supervised learning". In: *Advances in neural information processing systems*.

Grimm, Christopher, André Barreto, et al. (2020). "The Value Equivalence Principle for Model-Based Reinforcement Learning". In: *Advances in Neural Information Processing Systems*.

Grimm, Christopher, André Barreto, et al. (2021). "Proper Value Equivalence". In: *Advances in Neural Information Processing Systems*. Forthcoming.

Guan, Jonas et al. (2024). "Temporal-Difference Learning Using Distributed Error Signals". In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. URL: https://openreview.net/forum?id=8moTQjfqAV.

Guo, Zhaohan Daniel et al. (2022). "BYOL-Explore: Exploration by Bootstrapped Prediction". In: *Advances in Neural Information Processing Systems*.

Haarnoja, Tuomas et al. (2018a). "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *International Conference on Machine Learning*.

— (2018b). "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International Conference on Machine Learning*.

Hafner, Danijar et al. (2020). "Dream to Control: Learning Behaviors by Latent Imagination". In: *International Conference on Learning Representations*.

Hansen, Nicklas, Hao Su, and Xiaolong Wang (2024). "TD-MPC2: Scalable, Robust World Models for Continuous Control". In: *The Twelfth International Conference on Learning Representations*.

Hansen, Nicklas, Xiaolong Wang, and Hao Su (2022). "Temporal Difference Learning for Model Predictive Control". In: *International Conference on Machine Learning*.

Hansen, Nicklas A, Hao Su, and Xiaolong Wang (2022). "Temporal Difference Learning for Model Predictive Control". In: *International Conference on Machine Learning*.

Hasselt, Hado van (2010). "Double Q-learning". In: *Advances in Neural Information Processing Systems*.

Hasselt, Hado van, Arthur Guez, and David Silver (2016). "Deep reinforcement learning with double Q-Learning". In: *AAAI Conference on Artificial Intelligence*.

Heess, Nicolas et al. (2015). "Learning continuous control policies by stochastic value gradients". In: *Advances in neural information processing systems*.

Hiraoka, Takuya et al. (2022). "Dropout Q-Functions for Doubly Efficient Reinforcement Learning". In: *International Conference on Learning Representations*.

Hollenstein, Jakob et al. (2022). "Action Noise in Off-Policy Deep Reinforcement Learning: Impact on Exploration and Performance". In: *Transactions on Machine Learning Research*.

Hussing, Marcel et al. (2024). "Dissecting Deep RL with High Update Ratios: Combatting Value Divergence". In: *Reinforcement Learning Conference*.

Igl, Maximilian et al. (2021). "Transient Non-stationarity and Generalisation in Deep Reinforcement Learning". In: *International Conference on Learning Representations*.

Izenman, Alan Julian (1975). "Reduced-rank regression for the multivariate linear model". In: *Journal of multivariate analysis*.

Jaderberg, Max et al. (2016). "Reinforcement Learning with Unsupervised Auxiliary Tasks". In: *International Conference on Learning Representations*.

— (2017). "Reinforcement Learning with Unsupervised Auxiliary Tasks". In: *International Conference on Learning Representations*.

Janner, Michael et al. (2019a). "When to Trust Your Model: Model-Based Policy Optimization". In: *Advances in Neural Information Processing Systems*.

— (2019b). "When to Trust Your Model: Model-Based Policy Optimization". In: *Advances in Neural Information Processing Systems*.

Jin, Chi et al. (2020). "Provably efficient reinforcement learning with linear function approximation". In: *Conference on Learning Theory*.

Joseph, Joshua et al. (2013). "Reinforcement learning with misspecified model classes". In: *IEEE International Conference on Robotics and Automation*.

Kaplan, Jared et al. (2020). "Scaling laws for neural language models". In: *arXiv preprint arXiv:2001.08361*.

Kearns, Michael and Satinder Singh (2002). "Near-optimal reinforcement learning in polynomial time". In: *Machine learning* 49.2.

Kemertas, Mete and Tristan Ty Aumentado-Armstrong (2021). "Towards Robust Bisimulation Metric Learning". In: *Advances in Neural Information Processing Systems*.

Kim, Woojun et al. (2023). "Sample-Efficient and Safe Deep Reinforcement Learning via Reset Deep Ensemble Agents". In: *Advances in Neural Information Processing Systems*.

Kingma, Diederik and Jimmy Ba (2015). "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations (ICLR)*. San Diega, CA, USA.

Krogh, Anders and John Hertz (1991). "A Simple Weight Decay Can Improve Generalization". In: *Advances in Neural Information Processing Systems*.

Kumar, Aviral et al. (2021). "Implicit Under-Parameterization Inhibits Data-Efficient Deep Reinforcement Learning". In: *International Conference on Learning Representations*.

Kuznetsov, Arsenii et al. (2020). "Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics". In: *International Conference on Machine Learning*.

Laidlaw, Cassidy, Stuart Russell, and Anca Dragan (2023). "Bridging RL Theory and Practice with the Effective Horizon". In: *Thirty-seventh Conference on Neural Information Processing Systems*.

Lambert, Nathan et al. (2020). "Objective Mismatch in Model-based Reinforcement Learning". In: *Conference on Learning for Dynamics and Control*.

Lan, Qingfeng et al. (2020). “Maxmin Q-learning: Controlling the Estimation Bias of Q-learning”. In: *International Conference on Learning Representations*.

Laskin, Michael, Aravind Srinivas, and Pieter Abbeel (2020). “CURL: Contrastive Unsupervised Representations for Reinforcement Learning”. In: *International Conference on Machine Learning*.

Lavoie, Samuel et al. (2023). “Simplicial Embeddings in Self-Supervised Learning and Downstream Classification”. In: *International Conference on Learning Representations*.

Le Lan, Charline, Marc G Bellemare, and Pablo Samuel Castro (2021). “Metrics and continuity in reinforcement learning”. In: *AAAI Conference on Artificial Intelligence*, pp. 8261–8269.

Le Lan, Charline, Stephen Tu, Adam Oberman, et al. (2022). “On the Generalization of Representations in Reinforcement Learning”. In: *International Conference on Artificial Intelligence and Statistics*.

Le Lan, Charline, Stephen Tu, Mark Rowland, et al. (2023). “Bootstrapped Representations in Reinforcement Learning”. In: *International Conference on Machine Learning*.

Lee, Donghun, Boris Defourny, and Warren Buckler Powell (2013). “Bias-corrected Q-learning to control max-operator bias in Q-learning”. English (US). In: *Proceedings of the 2013 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, ADPRL 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI 2013*. IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning, ADPRL, pp. 93–99. ISBN: 9781467359252. DOI: 10.1109/ADPRL.2013.6614994.

Lee, Kimin et al. (2021). “Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning”. In: *International Conference on Machine Learning*. PMLR.

Levine, Sergey and Vladlen Koltun (2013). “Guided policy search”. In: *International Conference on Machine Learning*.

Li, Qiyang et al. (2023). “Efficient Deep Reinforcement Learning Requires Regulating Overfitting”. In: *International Conference on Learning Representations*.

Lillicrap, Timothy P et al. (2016). “Continuous control with deep reinforcement learning”. In: *International Conference on Learning Representations*.

Liu, Zhuang et al. (2021). “Regularization Matters in Policy Optimization - An Empirical Study on Continuous Control”. In: *International Conference on Learning Representations*.

Lovatto, Ângelo G. et al. (2020). “Decision-Aware Model Learning for Actor-Critic Methods: When Theory Does Not Meet Practice”. In: *”I Can’t Believe It’s Not Better!” at NeurIPS Workshops*.

Lu, Cong et al. (2024). “Synthetic experience replay”. In: *Advances in Neural Information Processing Systems*.

Luo, Yuping et al. (2019). "Algorithmic Framework for Model-based Deep Reinforcement Learning with Theoretical Guarantees". In: *International Conference on Learning Representations*.

Lyle, Clare, Mark Rowland, and Will Dabney (2022). "Understanding and Preventing Capacity Loss in Reinforcement Learning". In: *International Conference on Learning Representations*.

Lyle, Clare, Mark Rowland, Will Dabney, et al. (2022). "Learning Dynamics and Generalization in Deep Reinforcement Learning". In: *International Conference on Machine Learning*.

Lyle, Clare, Mark Rowland, Georg Ostrovski, et al. (2021). "On the effect of auxiliary tasks on representation dynamics". In: *International Conference on Artificial Intelligence and Statistics*.

Lyle, Clare, Zeyu Zheng, Khimya Khetarpal, et al. (2024). *Disentangling the Causes of Plasticity Loss in Neural Networks*. arXiv: 2402.18762 [cs.LG].

Lyle, Clare, Zeyu Zheng, Evgenii Nikishin, et al. (2023). "Understanding Plasticity in Neural Networks". In: *International Conference on Machine Learning*.

Madry, Aleksander et al. (2018). "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *International Conference on Learning Representations*.

Maei, Hamid et al. (2009). "Convergent temporal-difference learning with arbitrary smooth function approximation". In: *Advances in neural information processing systems* 22.

Mahadevan, Sridhar (2009). "Learning representation and control in Markov decision processes: New frontiers". In: *Foundations and Trends® in Machine Learning*.

Mannor, Shie et al. (Feb. 2007). "Bias and Variance Approximation in Value Function Estimates". In: *Manage. Sci.* 53.2, pp. 308–322.

Mikolov, Tomas et al. (2013). "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems*.

Mnih, Volodymyr et al. (2013). "Playing Atari With Deep Reinforcement Learning". In: *NeurIPS Deep Learning Workshop*.

Modi, Aditya et al. (2020). "Sample Complexity of Reinforcement Learning using Linearly Combined Model Ensembles". In: *International Conference on Artificial Intelligence and Statistics*.

Moerland, Thomas M. et al. (2023). "Model-based Reinforcement Learning: A Survey". In: *Foundations and Trends in Machine Learning* 16.1.

Moskovitz, Ted et al. (2021). "Tactical optimism and pessimism for deep reinforcement learning". In: *Advances in Neural Information Processing Systems*.

Nair, Suraj, Silvio Savarese, and Chelsea Finn (2020). "Goal-aware prediction: Learning to model what matters". In: *International Conference on Machine Learning*.

Nair, Vinod and Geoffrey E Hinton (2010). "Rectified linear units improve restricted boltzmann machines". In: *ICML 2010*, pp. 807–814.

Nauman, Michal, Michał Bortkiewicz, et al. (2024). "Overestimation, Overfitting, and Plasticity in Actor-Critic: the Bitter Lesson of Reinforcement Learning". In: *Forty-first International Conference on Machine Learning*.

Nauman, Michal, Mateusz Ostaszewski, et al. (2024). "Bigger, Regularized, Optimistic: scaling for compute and sample-efficient continuous control". In: *Advances in Neural Information Processing Systems*.

Ni, Tianwei et al. (2024). "Bridging State and History Representations: Understanding Self-Predictive RL". In: *To appear in International Conference on Learning Representations*.

Nikishin, Evgenii, Romina Abachi, et al. (2022). "Control-oriented model-based reinforcement learning with implicit differentiation". In: *AAAI Conference on Artificial Intelligence*.

Nikishin, Evgenii, Junhyuk Oh, et al. (2024). "Deep reinforcement learning with plasticity injection". In: *Advances in Neural Information Processing Systems* 36.

Nikishin, Evgenii, Max Schwarzer, et al. (2022). "The Primacy Bias in Deep Reinforcement Learning". In: *International Conference on Machine Learning*.

Oh, Junhyuk, Satinder Singh, and Honglak Lee (2017). "Value prediction network". In: *Advances in neural information processing systems* 30.

Ostrovski, Georg, Pablo Samuel Castro, and Will Dabney (2021). "The difficulty of passive learning in deep reinforcement learning". In: *Advances in Neural Information Processing Systems*.

Pan, Yangchen et al. (2019). "Hill climbing on value estimates for search-control in Dyna". In: *International Joint Conference on Artificial Intelligence*.

Peer, Oren et al. (2021). "Ensemble Bootstrapping for Q-Learning". In: *International Conference on Machine Learning*.

Pendrith, Mark and Malcolm Ryan (1997). "Estimator Variance in Reinforcement Learning: Theoretical Problems and Practical Solutions". In.

Pomerleau, Dean A. (1988). "ALVINN: An Autonomous Land Vehicle in a Neural Network". In: *Advances in Neural Information Processing Systems*.

Precup, Doina, Richard S. Sutton, and Sanjoy Dasgupta (2001). "Off-Policy Temporal Difference Learning with Function Approximation". In: *International Conference on Machine Learning*.

Pretorius, Arnu, Steve Kroon, and Herman Kamper (2018). "Learning dynamics of linear denoising autoencoders". In: *International Conference on Machine Learning*.

Puterman, Martin L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. 1st. USA: John Wiley & Sons, Inc. ISBN: 0471619779.

Rakhsha, Amin et al. (2022). "Operator Splitting Value Iteration". In: *Advances in Neural Information Processing Systems*.

Ross, Stéphane and Drew Bagnell (2012). "Agnostic System Identification for Model-Based Reinforcement Learning". In: *International Conference on Machine Learning*.

Saglam, Baturay et al. (2021). "Estimation Error Correction in Deep Reinforcement Learning for Deterministic Actor-Critic Methods". In: *2021 IEEE 33rd International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 137–144. DOI: 10.1109/ICTAI52525.2021.00027.

Saxe, Andrew M., James L. McClelland, and Surya Ganguli (2014). "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks". In: *International Conference on Learning Representations*.

Schneider, Jeff G (1997). "Exploiting model uncertainty estimates for safe dynamic control learning". In: *Advances in Neural Information Processing Systems*.

Schrittwieser, Julian et al. (2020). "Mastering atari, go, chess and shogi by planning with a learned model". In: *Nature* 588.7839.

Schwarzer, Max, Ankesh Anand, et al. (2021). "Data-Efficient Reinforcement Learning with Self-Predictive Representations". In: *International Conference on Learning Representations*.

Schwarzer, Max, Johan Samir Obando Ceron, et al. (2023). "Bigger, Better, Faster: Human-level Atari with human-level efficiency". In: *International Conference on Machine Learning*.

Schwarzer, Max, Nitarshan Rajkumar, et al. (2021). "Pretraining Representations for Data-Efficient Reinforcement Learning". In: *Advances in Neural Information Processing Systems*.

Silver, David, Hado van Hasselt, et al. (2017). "The predictron: end-to-end learning and planning". In: *International Conference on Machine Learning*.

Silver, David, Guy Lever, et al. (2014). "Deterministic policy gradient algorithms". In: *International Conference on Machine Learning*.

Sokar, Ghada et al. (2023). "The dormant neuron phenomenon in deep reinforcement learning". In: *International Conference on Machine Learning*.

Srivastava, Nitish et al. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15.56, pp. 1929–1958.

Sutton, Richard S (1988). "Learning to predict by the methods of temporal differences". In: *Machine learning*.

— (1990). "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming". In: *Machine learning Proceedings*.

Sutton, Richard S, A Rupam Mahmood, and Martha White (2016). "An emphatic approach to the problem of off-policy temporal-difference learning". In: *Journal of Machine Learning Research* 17.73, pp. 1–29.

Sutton, Richard S. and Andrew G. Barto (2018a). *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: A Bradford Book.

— (2018b). *Reinforcement Learning: An Introduction*. Second. The MIT Press.

Talvitie, Erin (2017). "Self-correcting models for model-based reinforcement learning". In: *AAAI Conference on Artificial Intelligence*.

Talvitie, Erin J et al. (2024). "Bounding-Box Inference for Error-Aware Model-Based Reinforcement Learning". In: *Reinforcement Learning Journal*.

Tang, Yunhao, Zhaohan Daniel Guo, et al. (2022). "Understanding Self-Predictive Learning for Reinforcement Learning". In: *International Conference on Machine Learning*.

Tang, Yunhao and Rémi Munos (2023). "Towards a better understanding of representation dynamics under TD-learning". In: *International Conference on Machine Learning*.

Tarasov, Denis et al. (2023). "Revisiting the Minimalist Approach to Offline Reinforcement Learning". In: *Advances in Neural Information Processing Systems*.

Thrun, Sebastian and Anton Schwartz (1993). "Issues in Using Function Approximation for Reinforcement Learning". In: *Proceedings of the 1993 Connectionist Models Summer School*.

Tian, Yuandong, Xinlei Chen, and Surya Ganguli (2021). "Understanding self-supervised learning dynamics without contrastive pairs". In: *International Conference on Machine Learning*.

Tomar, Manan et al. (2023). "Learning Representations for Pixel-based Control: What Matters and Why?" In: *Transactions on Machine Learning Research*.

Tsitsiklis, John and Benjamin Van Roy (1996). "Analysis of Temporal-Diffference Learning with Function Approximation". In: *Advances in Neural Information Processing Systems*.

Tunyasuvunakool, Saran et al. (2020a). "dm_control: Software and tasks for continuous control". In: *Software Impacts* 6, p. 100022. ISSN: 2665-9638. DOI: https://doi.org/10.1016/j.simpa.2020.100022.

— (2020b). "dm_control: Software and tasks for continuous control". In: *Software Impacts* 6.

Voelcker, Claas A, Arash Ahmadian, et al. (2024). *λ-models: Effective Decision-Aware Reinforcement Learning with Latent Models*. arXiv: 2306.17366 [cs.LG]. URL: https://arxiv.org/abs/2306.17366.

Voelcker, Claas A, Marcel Hussing, and Eric Eaton (2024). "Can we hop in general? A discussion of benchmark selection and design using the Hopper environment". In: *Finding the Frame: An RLC Workshop for Examining Conceptual Frameworks*. URL: https://openreview.net/forum?id=9IgtF63LPA.

Voelcker, Claas A, Marcel Hussing, Eric Eaton, et al. (2025). "MAD-TD: Model-Augmented Data stabilizes High Update Ratio RL". In: *International Conference on Learning Representations*.

Voelcker, Claas A, Tyler Kastner, et al. (2024). "When does self-prediction help? Understanding Auxiliary Tasks in Reinforcement Learning". In: *Reinforcement Learning Conference*.

Voelcker, Claas A, Victor Liao, et al. (2022). “Value Gradient weighted Model-Based Reinforcement Learning”. In: *International Conference on Learning Representations*.

Voelcker, Claas A, Anastasiia Pedan, et al. (2025). “Callibrated value-aware loss functions with stochastic environment models”. In: *under review*.

Wang, Che et al. (2020). “Striving for Simplicity and Performance in Off-Policy DRL: Output Normalization and Non-Uniform Sampling”. In: *International Conference on Machine Learning*.

Wang, Ziyu et al. (2016). “Dueling network architectures for deep reinforcement learning”. In: *International Conference on Machine Learning*.

Watkins, Christopher JCH and Peter Dayan (1992). “Q-learning”. In: *Machine learning* 8.3-4.

Watter, Manuel et al. (2015). “Embed to Control: A Locally Linear Latent Dynamics Model for Control from Raw Images”. In: *Advances in Neural Information Processing Systems*.

Wu, Dongming et al. (2020). “Reducing Estimation Bias via Triplet-Average Deep Deterministic Policy Gradient”. In: *IEEE Transactions on Neural Networks and Learning Systems*.

Xu, Jingjing et al. (2019). “Understanding and Improving Layer Normalization”. In: *Advances in Neural Information Processing Systems*.

Yang, Yuzhe et al. (2020). “Harnessing Structures for Value-Based Planning and Reinforcement Learning”. In: *International Conference on Learning Representations*.

Ye, Weirui et al. (2021). “Mastering atari games with limited data”. In: *Advances in Neural Information Processing Systems*.

Young, Kenny and Tian Tian (2019). “MinAtar: An Atari-Inspired Testbed for Thorough and Reproducible Reinforcement Learning Experiments”. In: *arXiv preprint arXiv:1903.03176*.

Zhang, Amy et al. (2021). “Learning Invariant Representations for Reinforcement Learning without Reconstruction”. In: *International Conference on Learning Representations*.

Zhang, Biao and Rico Sennrich (2019). “Root Mean Square Layer Normalization”. In: *Advances in Neural Information Processing Systems 32*. Vancouver, Canada.

Zhang, Zongzhang, Zhiyuan Pan, and Mykel J. Kochenderfer (2017). “Weighted Double Q-learning”. In: *International Joint Conference on Artificial Intelligence*.

Zhao, Yi et al. (2023). “Simplified Temporal Consistency Reinforcement Learning”. In: *International Conference on Machine Learning*.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean

faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit

erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.