Advances in Embodied Control: From Bayesian Exploration to Interactive Full-Body Motion Imitation

Kristian Hartikainen

Department of Computer Science
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

October 29, 2025

Abstract

Recent advancements in artificial intelligence, driven by machine learning, have revolutionized fields from computer vision to natural language processing. However, the physical capabilities of embodied agents, such as robots and virtual entities, still lag behind the agility of natural beings. This thesis explores the potential of learning-based methods and control theory to enhance the physical capabilities of artificial agents.

We develop Bayesian model-free RL approaches that quantify agent uncertainty, enabling more stable learning and improved exploration capabilities. We then explore the role of hierarchy and information asymmetry in model-free transfer learning, introducing a novel method for automating information asymmetry selection that enhances performance in simulated robot manipulation. Finally, we examine the use of keyframe motions as a source of priors and develop a model-predictive control method for interactive physics-based full-body motion imitation.

This thesis contributes novel algorithms, empirical analyses, and practical insights into the interplay between exploration, transfer learning, and motion imitation for embodied RL agents. These contributions collectively aim to advance the general field of RL and broaden the application spectrum of embodied agents in complex physical tasks.

Advances in Embodied Control: From Bayesian Exploration to Interactive Full-Body Motion Imitation

Kristian Hartikainen

Department of Computer Science

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

October 29, 2025

Acknowledgements

I feel fortunate to have had the support, guidance, and friendship of many individuals who made this thesis possible. First, I am thankful to Shimon for giving me the opportunity and freedom to explore my own path, and for his support throughout the journey.

I am grateful for my collaborators at Oxford. I am especially thankful to Mattie Fellows for introducing me to the Bayesian world, patiently answering countless questions, and generously hosting dinners and playing dance music for us. This thesis would not exist without your help. I'm also grateful to Sasha Salter for the invaluable partnership as we worked together and for the friendship that extended beyond Oxford. Together with Walter Goodwin, our collaboration was indispensable during the quieter days of COVID.

I also thank all my friends and colleagues at WhiRL for the inspiring discussions and supportive environment while it lasted. In particular, Tarun Gupta, Jake Beck, Max Igl, Luisa Zintgraf, Matthew Smith, Vitaly Kurin, Ben Ellis, Jelena Luketina, Shangtong Zhang, Wendelin Böhmer.

I am grateful to my collaborators and hosts at Google DeepMind: Leonard Hasenclever, Philemon Brakel, Steven Bohez, Wenhao Yu, Tingnan Zhang, Nicolas Heess, Taylor Howell, Yuval Tassa, Tom Erez, and Nimrod Gileadi. Working with you all in person during my internships gave me the environment that I much needed. I learned a great deal and might have taken a different path without those experiences.

Thank you to Nam Hee Kim, Perttu Hämäläinen, Elias Mikkola, and everyone else at Team ARAM at Aalto University for welcoming me into their research group, and for all the inspiring conversations that followed.

John, Greg, and Amy played a major role in shaping my early career. I have John to thank for many of my technical skills today, and I will always be grateful for the hospitality and inclusivity Greg and Amy showed me. Without them, I might have never explored beyond Finland.

I also want to thank Tuomas Haarnoja for a mentorship and collaboration that continues to this day. The overwhelmingly positive introduction to research I gained with you and Sergey at UC Berkeley was a major factor in my decision to pursue a PhD.

Life outside of work would have been a lot less fun without my friends, both old and new, near and far: Ats, Dann and Lauriina, Juuso and Eeva, Kenneth, Timo, Martti, Jarno, Avi, Connor and Anna, Anders, Kristian, and many others. You've made countless moments memorable. Special thanks to Risto for sharing so many thoughts, experiences, and interests — both personal and professional.

Finally, to my family — Mom, Dad, Tim, Eija, Alisa, Jussi, Kirsi, and Kapa — thank you for your wholehearted support and for giving me the opportunities that led me here. Your presence in my life has been invaluable.

And to Linda: thank you for your constant support, patience, and encouragement.

Abstract

Recent advancements in artificial intelligence, driven by machine learning, have revolutionized fields from computer vision to natural language processing. However, the physical capabilities of embodied agents, such as robots and virtual entities, still lag behind the agility of natural beings. This thesis explores the potential of learning-based methods and control theory to enhance the physical capabilities of artificial agents.

We develop Bayesian model-free RL approaches that quantify agent uncertainty, enabling more stable learning and improved exploration capabilities. We then explore the role of hierarchy and information asymmetry in model-free transfer learning, introducing a novel method for automating information asymmetry selection that enhances performance in simulated robot manipulation. Finally, we examine the use of keyframe motions as a source of priors and develop a model-predictive control method for interactive physics-based full-body motion imitation.

This thesis contributes novel algorithms, empirical analyses, and practical insights into the interplay between exploration, transfer learning, and motion imitation for embodied RL agents. These contributions collectively aim to advance the general field of RL and broaden the application spectrum of embodied agents in complex physical tasks.

Contents

List of Figures

List of Acronyms

APES *A*ttentive *P*riors for *E*xpressive and Transferable *S*kills. vi, vii, ix, x, 73, 75, 76, 85, 87, 89, 91–102, 146, 147, 156, 157, 159–166

BAC Bayesian Actor-Critic. ix, 66–68, 72

BBAC Bayesian Bellman Actor-Critic. ix, 67, 68, 70–72, 124, 139–142

BBO *B*ayesian *B*ellman *O*perators. vi, vii, ix, 38–40, 44–51, 53, 54, 56, 57, 60–64, 66, 68, 70, 71, 124, 126, 131, 133, 135–139

BootDQN+Prior Bootstrapped DQN+Prior. 40, 61, 65

ELBO Evidence Lower BOund. 15, 16

MAP Maximum A Posteriori. 13, 16, 17, 51–55, 62

MC Monte Carlo. 13, 14, 16, 27, 29, 30

RP-BBAC Randomized-Prior Bayesian Bellman Actor-Critic. ix, 58–61, 64–70, 142

RP Randomized Priors. 16, 53, 54, 56–58

SAC Soft Actor-Critic. ix, 40, 66–71, 87, 92, 93, 139, 140, 143, 151, 156

VI Variational Inference. 14–16, 147

BAMDP Bayes-adaptive Markov decision process. 40, 41, 45

MDP Markov decision process. 17–20, 22–26, 33–36, 41–43, 50, 53, 56, 58, 59, 62, 67

POMDP Partially observable Markov decision process. 34

RL Reinforcement Learning. iv, 1–6, 17, 19, 27, 28, 32, 40, 41, 43–45, 50, 53, 54, 60, 63, 65, 66, 68, 72, 89, 121, 122, 124, 125

1 Introduction

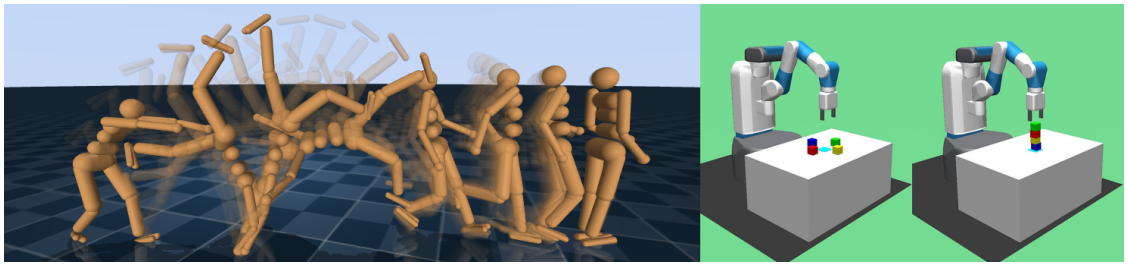


Figure 1.1: Physically-simulated agents performing motor control tasks using methods introduced in this thesis. (Left) Humanoid performing cartwheel from chapter 5. (Right) Robot manipulator stacking cubes from chapter 4.

The field of artificial intelligence has witnessed remarkable breakthroughs in recent years. Advances in machine learning have revolutionized a spectrum of domains, from computer vision to natural language processing. Our models now generate images [Karras et al., 2019, Mensah et al., 2023] and audio [Engel et al., 2020, Oord et al., 2016] that closely mimic reality, while in the realm of games, algorithms have achieved superhuman performance [Arulkumaran et al., 2019, Berner et al., 2019, Silver et al., 2017]. Furthermore, natural language processing has progressed from basic machine translation [Bahdanau et al., 2014] to creating human-resembling chat interfaces [Achiam et al., 2023, Team et al., 2023] and code from natural language [Brown et al., 2020, Chen et al., 2021]. These abilities are

all underpinned by the fundamental capacity to *learn* from data, adapt to new patterns, and make predictions based on the learned knowledge.

However, the remarkable cognitive progress in AI starkly contrasts with the capabilities of embodied artificial agents – robots or virtual entities capable of physical interaction with their environments. While AI agents showcase impressive intellectual feats, their physical abilities still lag far behind the agility and versatility exhibited by humans and other animals. We can perform complex motor skills in dynamic environments with stunning ease, a feat that still eludes our most advanced artificial agents.

Learning has the potential to transform the physical capabilities of AI, just as it has revolutionized its cognitive abilities. However, embodied intelligence poses a fundamentally different challenge. Physical movement is inherently sequential; our actions influence what we experience next. This starkly contrasts with traditional learning paradigms where data is often assumed to be independent and identically distributed. Supervised learning, while powerful, frequently proves inadequate for control of embodied agents. Instead, we must turn to other paradigms, such as Reinforcement Learning (RL), a framework specifically designed for sequential decision-making. Unlike supervised learning, RL is not about matching patterns but about learning to make sequences of decisions that maximize some notion of long-term returns. This involves not just recognizing patterns and predicting outcomes but also actively finding good solutions and understanding the causes for them.

In RL, we tell the agent *what* to do and the learning process must discover, through trial and error, *how* to achieve that goal. This is akin to teaching someone how to play chess by only telling them the objective (checkmate the opponent's king) but not the specific moves required to get there. The agent must *explore* the unknown environment, experiment with different actions, and learn from the resulting rewards or penalties to progressively improve its decision-making capabilities.

Exploration is a fundamental aspect of reinforcement learning, crucial for discovering effective strategies in complex environments. Unlike more passive learning methods, RL agents actively influence their environment. They must

balance between exploiting known actions with positive outcomes and exploring new actions that could yield even greater long-term benefits. This need arises because environments in RL are typically characterized by uncertainty and incomplete information. An agent acting greedily, choosing actions that yield the highest immediate reward, may miss out on discovering actions that could lead to much higher returns further in the future.

Exploration is ubiquitous in nature. For instance, a bird in a familiar forest might rely on known berry bushes for sustenance. However, a drought that withers these sources forces the bird to explore. Hunger drives it to venture further, seeking out unfamiliar trees and bushes. This search carries risk and expends energy, but the potential reward — finding new, edible berries — is essential for survival.

Biological agents rarely explore their environments unguided, however. Animals leverage past experiences and knowledge passed down through generations via genes to focus their exploration at the right level of abstraction, thereby reducing the time and energy spent on fruitless options. Imagine a baby bird. Even if starving, lacking any prior life experience, it would not be able to formulate the exploration problem that would lead to food. It is the transferred knowledge, in the bird's case, the ability to fly, that enables efficient exploration on the bush level; without prior knowledge, the exploration would happen in the wing-flapping level at best, never covering any of the berry bushes.

Transfer learning is a technique that aims to incorporate similar knowledge transfer capabilities into our artificial agents. The goal is to apply expertise gained in one domain to improve learning or performance in another, even a seemingly unrelated one. Within reinforcement learning, transfer learning allows agents to exploit knowledge gained in previous tasks. This can accelerate learning, reduce data requirements, and improve generalization or safety across tasks. By using patterns, strategies, or policies learned from one setting, agents can explore more efficiently and adapt faster to new environments or challenges. This provides a more versatile toolkit for tackling novel or complex tasks that might otherwise require extensive learning from scratch. Incorporating transfer learning into the

development of RL agents offers a powerful means to make them more capable and efficient by building on a wider range of experiences.

While learning-based approaches, like reinforcement learning, hold immense potential in addressing the challenges of embodied intelligence, it is important to acknowledge that they are not the exclusive solution. Traditional control methods, rooted in control theory, continue to play a pivotal role in tackling sequential decision-making problems. Control theory offers a rich set of techniques for designing systems with predictable and robust behavior when the dynamics of the environment are well understood. Especially in scenarios like simulated robotics and physics-based computer animation, where we have access to detailed knowledge about the underlying environment, control-theoretic methods can provide highly effective solutions. Moreover, with advancements in computing power and optimization techniques, control methods have become faster at handling complex systems. Integrating the strengths of both learning-based approaches and established control methods can be a powerful strategy for advancing the capabilities of embodied AI agents.

The ideal approach for controlling embodied agents is yet to be determined, as the field continues to actively explore these methods. Currently, there is no one-size-fits-all solution, and the most effective control strategy depends on the specific task and environment the agent is encountering. These advancements in embodied control offer enormous potential across various applications, with the ability to revolutionize several domains. In robotics, it could enable robots to assemble complex products in dynamic factory environments, navigate cluttered warehouses with precision, and provide in-home assistance with the flexibility to handle a multitude of everyday tasks. It could transform computer graphics and animation, empowering virtual characters to learn natural, nuanced motions autonomously. Additionally, embodied intelligence can reshape biomechanics and physiotherapy by allowing researchers and clinicians to simulate and analyze the impact of different interventions, ultimately leading to more personalized and effective treatments.

In this thesis, we explore the strengths of both learning-based methods and control theory for embodied intelligence. While earlier chapters delve into the potential of reinforcement learning and the benefits of transfer learning, we will also investigate scenarios where control methods still provide advantages over RL-based solutions.

1.1 Thesis Overview

This thesis is divided into six chapters. In chapter 2, we begin by introducing the relevant background concepts that we will build upon, offering a primer for readers unfamiliar with the field. The subsequent three chapters, chapters 3 to 5, contain the main contributions of this thesis.

In chapter 3, which is based on [Fellows et al., 2021], we explore a Bayesian approach to model-free reinforcement learning. Our goal is to develop practical methods for quantifying the agent's uncertainty about the environment. This enables more informed and deeper exploration, and ultimately, improved asymptotic performance compared to contemporary model-free algorithms.

Chapter 4, based on [Salter et al., 2020], investigates data-driven methods for knowledge transfer in model-free reinforcement learning. Focusing on hierarchical KL-regularized RL, we analyze how hierarchy and information asymmetry influence successful transfer. Furthermore, we propose a practical method for learning information asymmetries and demonstrate its effectiveness in simulated robot manipulation tasks.

In chapter 5, we shift our focus to more traditional control methods and alternative ways of synthesizing embodied behaviors. Specifically, we leverage keyframe motion data for humanoid full-body control. We present a practical method designed for interactive physics-based motion tracking and experimentally demonstrate its superiority over existing RL-based motion tracking methods.

Finally, we conclude the thesis in chapter 6, where we summarize key insights and discuss potential directions for future research.

1.2 Key Contributions

This thesis makes several methodological and algorithmic contributions to the field of reinforcement learning and physically-simulated motor control, particularly in the application of these methods to embodied agents. Here, we outline the key advancements presented in each chapter:

- Chapter 3 introduces novel model-free Bayesian algorithms for policy evaluation and continuous control. Empirical evidence demonstrates their convergence properties and superior exploration capabilities in practical scenarios.
- Chapter 4 emphasizes the crucial role of information asymmetry for successful skill transfer in KL-regularized model-free RL. We introduce the expressivity-transferability trade-off and provide empirical support for it. We further introduce a novel method for automating the selection of information asymmetry and demonstrate improved performance and adaptability of our method in a challenging simulated cube-stacking robot manipulation task.
- Chapter 5 presents a model-predictive control method for interactive physics-based full-body motion tracking. We expose limitations of existing RL-based motion tracking approaches. Experiments on a large-scale motion dataset demonstrate that the method outperforms RL-based solutions, establishing a strong baseline for future methods.

2
Background

Contents

2.1 Probability Theory and Uncertainty

Understanding uncertainty and making decisions under uncertain conditions are foundational to reinforcement learning methods central to this thesis. This section introduces the basic concepts of probability theory and Bayesian inference, which allow us to quantify and reason about uncertainty.

2.1.1 Probability Theory

One of the fundamental concepts in sequential decision making is that of uncertainty. Probability theory provides a formal framework for describing the behaviors and predicting the likelihood of various outcomes in uncertain conditions.

A *random variable*, denoted by capital letters like X, represents a quantity whose *outcome* is uncertain and assumes different values as a result of some random process. For example, X could represent the outcome of rolling a six-sided die, where each possible outcome (1 through 6) is a value that X can take. A *random event* is a specific outcome or a set of outcomes from a random process. For example, the event that a six-sided die shows a number greater than 4 is a random event, and encompasses the outcomes 5 and 6.

We are often interested in the probability that a value of a random variable X belongs to a set A. For example, in the above die example, we might be interested in the probability that $X = 3$, in which case $A = \{3\}$. Similarly, for $X > 4$, $A = \{5, 6\}$. The *probability distribution*, denoted by P_X or $P(X)$, is a set function that describes these probabilities. It is a rule that assigns a probability to each outcome in the space of possible outcomes. The above die events can be denoted as $P_X(X = 3) = P_X(X \in \{3\})$ and $P_X(X > 4) = P_X(X \in \{5, 6\})$.

Random variables can be classified into two types: discrete and continuous. *Discrete random variables* take on a finite or countably infinite number of distinct values, such as the outcome of rolling a die. The *probability mass function* (pmf), denoted as $p_X(x) = P(X = x)$, where $0 \leq p_X(x) \leq 1$, defines the probability distribution of a discrete random variable, assigning probabilities to each of its possible values. The sum of $p_X(x)$ across all possible values of X equals 1, i.e. $\sum_{x \in X} p_X(x) = 1$, ensuring that the total probability across all possible outcomes is 1.

Continuous random variables, on the other hand, take on an uncountable range of values, such as the measurements of height in a population where $x \in \mathbb{R}$. For such variables, the absolute likelihood of X assuming any singular value x is invariably 0. Instead, we use a *probability density function* (pdf), which describes how densely the probabilities are distributed around that value. The pdf, denoted as $p_X(x)$,

allows us to compute the relative likelihood of sample x, yielding finite probability values for continuous events. The probability that X falls within an interval $[a, b]$ is given by the integral of p_X over the interval: $P(a \leq X \leq b) = \int_a^b p_X(x)\,dx$. The pdf must satisfy two conditions: 1) $p_X(x) \geq 0\ \forall x \in X$ and 2) the total area under the curve integrates to 1, i.e. $\int_{-\infty}^{+\infty} p_X(x)\,dx = 1$.

The *cumulative distribution function* (CDF) gives the probability that a variable X takes a value less than or equal to a value x, and is defined as:

$$P(X \leq x) = \int_{-\infty}^{x} p(x')\,dx'.$$

Understanding the relationships between multiple random variables is crucial in many areas of probability theory and its applications. These relationships can be described using joint, marginal, and conditional probabilities.

Joint probability describes the likelihood of two or more events happening at the same time. For two random variables, X and Y, the joint probability that X equals x and Y equals y is denoted as $P(X = x, Y = y)$.

Marginal probability refers to the probability of an event occurring, regardless of the outcome of another variable. For continuous variables, the marginal probability $P(X = x)$ can be found by integrating the joint probability over all possible values of Y: $p(x) = \int p(x, y)\,dy$. For discrete variables, it involves summation instead of integration: $p(x) = \sum_y p(x, y)$.

Conditional probability quantifies the probability of one event given the occurrence of another event. It is represented as $P(X = x|Y = y)$, meaning the probability of $X = x$ given that $Y = y$ has occurred. This concept is fundamental to understanding how the knowledge of one variable affects the distribution of another.

Two variables, X and Y, are considered *independent* if the occurrence of one does not affect the probability of occurrence of the other, which mathematically means $P(X = x|Y = y) = P(X = x)$ for all values of x and y.

Bayes' rule is a fundamental theorem in probability theory that allows reordering of conditional probabilities. It is derived from the concepts of conditional probability and product rule for events A and B, written as $P(A, B) =$

$P(A)P(B|A) = P(B)P(A|B)$. Bayes' rule is stated by re-ordering these terms as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.1}$$

Another important concept is the *expectation* of a function f of a random variable X, denoted as $\mathbb{E}_{x\sim p(x)}[f(x)]$. The expectation provides a measure of the central tendency of the function's distribution and is defined as the weighted average of all possible values of $f(X)$, with weights corresponding to their probabilities. Mathematically, for a random variable X with density function p, the expectation of $f(X)$ is given by:

$$\mathbb{E}_{x\sim p(x)}[f(x)] = \int f(x)p(x)\,dx.$$

For discrete variables, the expectation is computed as a sum rather than an integral. We often simplify the expectation notation to $\mathbb{E}[\cdot]$ by omitting the subscript when the distribution is clear from the context.

When f is the identify function, that is $f(X) = X$, the expectation is known as the *expected value* or *mean* of X, denoted as $\mu = \mathbb{E}[X]$. The mean provides a summary measure of the central location of the distribution of X. The *variance* $\mathrm{Var}(X)$ of a random variable X measures the spread of X's distribution around its mean and is defined as the expectation of the squared deviation of X from its mean: $\mathrm{Var}(X) = \mathbb{E}[(X-\mu)^2]$. The square root of the variance is known as the *standard deviation* and denoted as $\sigma = \sqrt{\mathrm{Var}(X)}$.

A *conditional expectation*, denoted as $\mathbb{E}[X|Y=y]$, describes the expected value of X given that $Y = y$. It averages out the uncertainty in X while taking into account the known value of Y.

We use various information-theoretic concepts in this thesis. The *entropy* of a variable X is defined as $\mathcal{H}(X) = \mathbb{E}_{P_X}[-\log P(X)]$, and describes the average information provided by a random event. When the random variable is continuous, the entropy is also referred to as differential entropy.

Finally, the *Kullback-Leibler divergence* (or KL-divergence) provides a statistical distance measure between two probability distributions P and Q:

$$\mathrm{D}_{\mathrm{KL}}(P \parallel Q) \coloneqq \int \log\left(\frac{p(x)}{q(x)}\right) p(x)\, dx$$

We sometimes abuse the notation of functions like KL-divergence and apply them to probability densities instead of probability distributions.

2.1.2 Bayesian Inference

Bayesian epistemology has recently gained increasing interest in machine learning and reinforcement learning. A Bayesian perspective involves using some observed data and our understanding of how the world works (our model) to update our beliefs about uncertain quantities. This updating process results in what is known as a posterior distribution, which quantifies our updated beliefs about the values of the uncertain quantities after considering the new evidence.

Concretely, in the inference problem, we have access to a dataset $\mathcal{D} = \{x_1, \ldots, x_N\}$ consisting of N independent and identically distributed (i.i.d.) *observations* of a random variable X, and assume that the dataset generation was influenced by an *unobserved latent* variable Z. Given the observed dataset $\mathcal{D}$, our goal is to *infer* the conditional probability, called *posterior* distribution $P_Z(\cdot|\mathcal{D})$ with density $p(z|\mathcal{D})$, of the latent Z. We can infer the posterior density using the Bayes' rule from equation (2.1), which now takes the form:

$$p(z|\mathcal{D}) = \frac{p(\mathcal{D}|z)p(z)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|z)p(z)}{\int p(\mathcal{D}|z)p(z)\, dz} \tag{2.2}$$

Here, $p(z)$ denotes the density of our *prior* belief of z, capturing the assumptions about z before observing the dataset $\mathcal{D}$. $p(\mathcal{D}|z)$ denotes the *likelihood* density of the observed data under our chosen model and given latent z, expressing how likely the observed dataset is for different values of the latent variable z. $p(\mathcal{D})$ denotes the *marginal likelihood*, also known as the *evidence*, of $\mathcal{D}$.

For example, consider the task of determining whether a die is fair. We roll the die N times, resulting in a sequence of samples in the range $\{1, ..., 6\}$, represented

as our dataset $\mathcal{D}$, where each element is an observation from the variable (values from 1 through 6) representing the outcome of a single die roll. The latent variable, Z, which we aim to infer, is the die's bias, representing the probabilities of each event. Using Bayesian inference, we update our beliefs about Z based on the observed data, computing the posterior distribution $P(Z|\mathcal{D})$ from our prior belief about Z and the likelihood of observing $\mathcal{D}$ given Z.

Bayesian modeling utilizes the posterior distribution to predict unseen data, employing what is called the *posterior predictive distribution* with density:

$$p(\mathcal{D}^*|\mathcal{D}) = \int p(\mathcal{D}^*|z)p(z|\mathcal{D})dz, \tag{2.3}$$

This integral, also known as the Bayesian model average, averages predictions from all plausible models weighted by their posterior probability. It formally characterizes our uncertainty regarding variables dependent on unknown latents. Unlike frequentist statistics, Bayesian solutions depend on the choice of prior, meaning that agents under different prior beliefs might infer different posterior beliefs from identical data with finite samples. Yet, it can be shown that, under certain regularity assumptions and in the limit of infinite samples, these beliefs will concentrate on the Maximum Likelihood Estimator (MLE)

$$z_{\text{MLE}} = \arg\max_z p(\mathcal{D}|z)$$

regardless of the agent's initial choice of prior [Van der Vaart, 2000].

While the equations above convey a sense of simplicity in the Bayesian inference process, they belie the complexity encountered with most real-world applications. In scenarios requiring straightforward models — such as linear approximators or Gaussian models — the analytical solution to these equations is achievable. However, for many problems of our interest, the task of approximating integrals in equations (2.2) and (2.3) becomes intractable. This complexity is particularly pronounced with high-dimensional latent variables and when employing function approximation techniques.

To overcome these challenges, one might resort to the Maximum A Posteriori (MAP) estimate as a simpler Bayesian approximation method. The MAP estimate is obtained by finding the most probable value of the hidden variables maximizing the posterior distribution:

$$z_{\text{MAP}} = \arg\max_{z} p(z|\mathcal{D}).$$

Although MAP is computationally more tractable than the full posterior, it reduces the posterior to a point estimate, losing the uncertainty that Bayesian methods aim to preserve.

There exists extensive literature dedicated to approximating the "fully Bayesian" solutions that compute the integral with respect to and provide uncertainty estimates of unknown variables for large datasets and complex models. We provide a short primer on these approaches in the following.

2.1.3 Monte Carlo Sampling

Monte Carlo (MC) sampling methods are a broad class of computational algorithms for approximating probability distributions and performing integrations in settings where analytical solutions are infeasible or intractable. This approach is particularly beneficial in the context of Bayesian inference, where we often encounter high-dimensional integrals in the computation of posterior distributions and expectations. The fundamental premise of MC sampling is to use randomly drawn samples to approximate these complex distributions or to estimate integrals.

Consider the problem of estimating the expectation of a function $f(x)$ under a probability distribution with density $p(x)$. Its expectation is defined as $\mathbb{E}[f(x)] = \int f(x)p(x)\,dx$ and, in many practical scenarios, solving the integral directly is challenging. Monte Carlo integration offers a solution by approximating this expectation with the sample mean from N samples $\{x_i\}_{i=1}^{N}$ drawn from the distribution with density $p(x)$:

$$\mathbb{E}[f(x)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i). \tag{2.4}$$

The accuracy of this approximation improves with the increase in the number of samples N, according to the law of large numbers, making MC sampling *consistent*. Another key property of the MC estimator is that it is *unbiased*, meaning that the expected value of the MC estimate equals the true value of the integral it approximates.

The application of Monte Carlo sampling extends beyond simple expectation estimation to include more sophisticated techniques such as Markov Chain Monte Carlo (MCMC) and Importance Sampling, which are designed to efficiently sample from complex distributions that are not amenable to direct sampling. These methods further expand the utility of MC sampling in Bayesian inference by facilitating the approximation of posterior distributions and enabling the computation of Bayesian model averages in scenarios where direct analytical solutions are not feasible.

Despite its many benefits, Monte Carlo sampling has drawbacks, particularly when dealing with high-dimensional spaces or distributions that are difficult to sample from efficiently. The variance of the MC estimator can be high, especially with a small number of samples, leading to potentially unstable or inaccurate estimates. Additionally, in complex models, drawing samples directly from the posterior distribution can be computationally challenging.

These limitations motivate the exploration of alternative approaches, such as Variational Inference (VI), which offers a different strategy by turning the inference problem into an optimization problem.

2.1.4 Variational Inference

Unlike Monte Carlo methods, which approximate distributions using samples, VI posits a family of distributions (often simpler and more tractable than the target distribution) and finds the member of this family that is closest to the target distribution. This approach is particularly effective for complex models where exact posterior computation is intractable.

The core of VI is to approximate the true posterior distribution $P_Z(\cdot|X)$ with a simpler, parameterized distribution Q_θ with density $q_\theta(z)$, where θ denotes the

parameters of the approximate distribution. The choice of Q_θ is crucial and is often guided by the trade-off between computational tractability and the ability to closely approximate the true posterior. The discrepancy between the true posterior density $p(z|x)$ and the approximation $q_\theta(z)$ is measured using a statistical distance measure. While various tractable statistical distance measures can be employed based on the specifics of the problem and computational considerations, the Kullback-Leibler (KL) divergence is commonly used.

The objective of VI is to minimize this divergence, effectively making the approximate density $q_\theta(z)$ as close as possible to the true posterior density $p(z|x)$:

$$\theta^* = \arg\min_\theta \mathrm{D}_{\mathrm{KL}}(Q_\theta \parallel P_Z(\cdot|X)). \tag{2.5}$$

However, directly minimizing $\mathrm{D}_{\mathrm{KL}}(Q_\theta \parallel P_Z(\cdot|X))$ is often infeasible because it involves the intractable posterior $p(Z|X)$. Instead, VI often focuses on maximizing the Evidence Lower BOund (ELBO), which is equivalent to minimizing the KL divergence. The ELBO can be formulated by expanding the KL-divergence:

$$\begin{aligned}
\mathrm{D}_{\mathrm{KL}}(Q_\theta \parallel P_Z(\cdot|X)) &= \mathbb{E}_{q_\theta(Z)}\left[\log q_\theta(Z|X)\right] - \mathbb{E}_{q_\theta(Z)}\left[\log p(Z|X)\right] \\
&= \mathbb{E}_{q_\theta(Z)}\left[\log q_\theta(Z|X)\right] - \mathbb{E}_{q_\theta(Z)}\left[\log p(Z,X)\right] + \log p(X) \\
&= -\mathrm{ELBO}(\theta) + \log p(X)
\end{aligned} \tag{2.6}$$

and ELBO can then be written as:

$$\mathrm{ELBO}(\theta) = \log p(X) - \mathrm{D}_{\mathrm{KL}}(Q_\theta \parallel P_Z(\cdot|X)) \leq \log p(X) \tag{2.7}$$

With the final lower bound following from the non-negativity of the KL-divergence. The ELBO can be understood as a balance between fitting the model to the data (first term) and regularizing the complexity of the model (second term). While the optimization in equation (2.5) is infeasible, maximizing the ELBO with respect to θ provides a tractable way to approximate the posterior distribution under some often satisfied assumptions. This optimization reduces the distance between $q_\theta(Z|X)$ and $p(Z|X)$, and is typically performed using iterative

gradient-based methods, which require computing the gradients of the ELBO with respect to the variational parameters θ.

VI offers several advantages, including scalability to large datasets and the ability to use complex, expressive families of distributions for the approximation. However, it is not without drawbacks. The choice of the variational family $q_\theta(Z)$ can limit the accuracy of the approximation. Moreover, VI introduces a bias due to the approximation, in contrast to the unbiased nature of Monte Carlo sampling. Despite these limitations, VI remains a powerful, widely used tool for approximate Bayesian inference, especially in complex models where exact inference is computationally prohibitive.

2.1.5 Randomized Priors

Randomized Priors (RP) presents an alternative approach to approximate inference by injecting noise into the MAP point estimate through the prior distribution [Fellows, 2021, Osband et al., 2018]. This provides a way to approximate an intractable posterior distribution while also accounting for uncertainty. While randomized priors are motivated in the context of linear Gaussian models, in which the sampling is exact, they provide good posterior approximations even with nonlinear neural networks [Fellows, 2021, Osband et al., 2018, Pearce et al., 2019].

The approximate posterior can be expressed as [Fellows, 2021]:

$$P_\Phi(\cdot|\mathcal{D}_N) \approx \hat{P}_\Phi(\cdot|\mathcal{D}_N) = \int \delta_{g_N(\epsilon)} dP_E(\epsilon),$$

where P_E represents the prior distribution and $g_N(\epsilon)$ is the solution to the prior-randomized MAP optimization problem defined as:

$$g_N(\epsilon) = \arg\max_\phi \log(p(\mathcal{D}_N|\phi)p_\Phi(\phi - \epsilon)). \tag{2.8}$$

To obtain a sample from the approximate posterior, we can first draw a sample $\epsilon \sim P_E$ from the prior distribution P_E and then find the MAP estimate in equation (2.8). Since finding the exact MAP solution is intractable for nonlinear models, we often use an ensemble approach along with Monte Carlo sampling.

For a test function $f(\phi)$ and a set of L samples $\epsilon_l \sim P_E$ from the prior, we can approximate the posterior as follows:

$$\int f(\phi)\, d\hat{P}_\Phi(\phi|\mathcal{D}_N) = \int f \circ g_N(\epsilon)\, d\hat{P}_E(\epsilon) \approx \frac{1}{L}\sum_{l=1}^{L} f \circ g_N(\epsilon_l) = \frac{1}{L}\sum_{l=1}^{L} f(\phi_l),$$

where each $\phi_l \in g_N(\epsilon_l)$ is a solution to the corresponding prior-randomized MAP optimization problem. Maintaining an ensemble of L optimal parameters, $\{\phi_l\}_{l=1}^{L}$, thus gives us a practical and tractable way to estimate the posterior.

2.2 Sequential Decision Making and Reinforcement Learning

In the sequential decision making problem, a decision maker encounters a sequence of system states and is tasked with making informed decisions so as to control the system's evolution towards desired outcomes. The process is inherently sequential, as the outcomes observed later are directly influenced by earlier decisions and states. This thesis considers two closely related frameworks that formalize this problem: Reinforcement Learning (RL) and control theory. Both of these frameworks tackle the complexity of the sequential decision making problem setting by separating the system into a formal interface between the *environment* — which defines the possible system states, available decisions, and the consequent outcomes of those decisions — and the *agent* — the decision-making entity that interacts with and aims to influence the system. These frameworks are highly general and find application across a vast array of practical decision-making problems.

The control of embodied agents, where an agent interacts with a physical or physically simulated environment, serves as the primary focus of this thesis. RL and control theory provide perspectives for addressing the challenges in embodied control, centered around the Markov decision process (MDP) formalism, which we describe in more detail next.

2.2.1 Markov Decision Processes

The Markov decision process (MDP) is a fundamental framework for describing the environment in which a decision-making agent operates. It offers a structured way to represent scenarios, accommodating a wide range of real-world settings while offering flexibility in how states, tasks, and actions are represented.

An MDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, P, P_0, r, \gamma)$. The state space $\mathcal{S}$ denotes the set of possible environment states and the action space $\mathcal{A}$ the set of actions (or controls) that the agent can choose from. $\mathcal{S}$ and $\mathcal{A}$ can in general be either discrete or continuous but, as common in embodied learning and control, we assume both to be continuous. The initial state distribution in the environment is denoted with P_0 and its density function with $p_0(\mathbf{s}_0)$.

In a sequence of a possibly infinite number of discrete time steps t, the agent observes a state $\mathbf{s}_t \in \mathcal{S}$ and acts in the environment by choosing an $\mathbf{a}_t \in \mathcal{A}$. Upon each time step, the environment transitions into a new state $\mathbf{s}_{t+1} \sim P(\cdot|\mathbf{s}_t, \mathbf{a}_t)$ according to environment dynamics governed by the state transition distribution P with the corresponding density function $p: \mathcal{S} \times \mathcal{S} \times \mathcal{A} \to [0, \infty)$. The reward function is denoted by $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [r_{\min}, r_{\max}]$ and on each transition, the environment emits a bounded reward $r_t \coloneqq r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})$, with $-\infty < r_{\min} \leq r_{\max} < +\infty$, which provides a signal for the agent to guide its decision making.

In the following, we use shorthands such as $\mathbf{s}_{\leq t} = \mathbf{s}_{0:t} = (\mathbf{s}_0, \ldots, \mathbf{s}_t)$ to denote states up to time step t, $\mathbf{a}_{<t} = \mathbf{a}_{0:t-1} = (\mathbf{a}_0, \ldots, \mathbf{a}_{t-1})$ to denote actions before time step t. τ_N denotes a state-action trajectory up to N time steps $\tau_N \coloneqq (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_{N-1}, \mathbf{a}_{N-1}, \mathbf{s}_N)$ and τ to denote an infinite length trajectory $\tau \coloneqq \lim_{N\to\infty} \tau_N = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \ldots)$.

Notably, the standard MDP formulation satisfies the *Markov assumption*, which posits that the future state distribution relies solely on the current state and action, independent of the sequence of events that preceded it. Simplifying decision-making, this property ensures that considering the current state and action is sufficient for predicting future outcomes, without the need for historical data.

Sequential decision making, as the name suggests, inherently focuses on optimizing not just the immediate reward but rather an aggregate of rewards over time, referred to as *return*. The concept of returns in RL and control theory is fundamental to understanding how agents optimize their policy to achieve long-term objectives. The formulation of return can vary, each variation offering distinct advantages for different scenarios. This work considers two commonly used settings: finite horizon and discounted.

The *finite horizon* setting, often referred to as the *receding horizon* in control theory, offers a straightforward approach to optimizing cumulative rewards over a fixed period. The return in a finite horizon scenario is defined as the sum of rewards over the first H time steps:

$$R(\tau) = \sum_{t=0}^{H} r_t. \tag{2.9}$$

In reinforcement learning, this approach often corresponds to episodic learning, where the agent's environment resets after H time steps, and a new episode begins with an initial state sampled from P_0. This explicit horizon facilitates planning over manageable future intervals without needing to consider indefinitely distant rewards. As explored in chapter 5, this setting bears close similarity to the model-predictive controllers used in control theory. In such a setting, instead of resetting the environment after H time steps, the receding horizon problem is used to repeatedly plan over a shorter horizon while sliding the fixed-span window along the time axis so as to control the system over a longer horizon.

In contrast, *discounted* MDPs introduce a discount factor $\gamma \in [0, 1)$ to prioritize immediate rewards over future rewards in an infinite horizon setting:

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t. \tag{2.10}$$

The discount factor γ can be seen as representing the continuity of the agent's interaction with the environment, with lower values indicating a higher likelihood of

terminating its operation at each step, after which the environment is reset [Sutton and Barto, 2018]. This approach not only reflects a pragmatic consideration of uncertainty in an agent's operation but also aligns with natural decision-making processes, where immediate outcomes often bear more significance than distant ones.

In this work, we are mostly concerned with the γ-discounted formulation, although the model-predictive control system in chapter 5 makes use of the finite-horizon setting. The rest of this section also presents things assuming the discounted setting.

2.2.2 Policies

So far we have introduced the interface for the environment in which our decision maker operates. Next, we will describe the *agent* and how to learn decision making policies so as to solve the MDPs.

We represent the agent's decision making *policy* as $\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, expressing the probability density of choosing an action $\mathbf{a} \sim \pi(\mathbf{a}|\mathbf{s})$ at a given environment state $\mathbf{s}$. The distribution of trajectories τ_N induced by the policy π, together with the environment dynamics, is denoted as P_N^τ. Density for these policy-induced trajectories is defined as $p_N^\pi(\tau_N) \coloneqq p_0(\mathbf{s}_0) \prod_{t=0}^{N-1} \pi(\mathbf{a}_t|\mathbf{s}_t) p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$, extending to infinite-length trajectories as $p^\pi(\tau) \coloneqq \lim_{N\to\infty} p_N^\pi(\tau)$.

Ultimately, the objective in reinforcement learning is to find an optimal policy that maximizes the expected future return objective:

$$\pi^* = \arg\max_\pi J^\pi, \tag{2.11}$$

$$\text{with } J^\pi = \mathop{\mathbb{E}}_{\tau\sim p^\pi(\tau)} [R(\tau)] . \tag{2.12}$$

Most of this work considers parametric policies, in which case we denote the policy with parameters θ as π_θ and the optimization objective becomes a function of the policy parameters:

$$\pi^* = \arg\max_\theta J(\theta), \tag{2.13}$$

$$\text{with } J(\theta) = \mathop{\mathbb{E}}_{\tau\sim p^\pi(\tau)} [R(\tau)] . \tag{2.14}$$

In cases assuming parameterized policies, we assume that the policy function π_θ is differentiable with respect to its parameters θ.

2.2.3 Value Functions

To facilitate discussion of strategies for learning optimal policies, we first introduce the concept of value functions. Value functions are a central concept in reinforcement learning, providing estimates of the policy's quality in the form of expected future returns. The *state-value function*, $V^\pi(\mathbf{s}) : \mathcal{S} \to \mathbb{R}$, of a policy π captures the expected return from starting in state $\mathbf{s}$ and following π:

$$V^\pi(\mathbf{s}) = \mathbb{E}_{\tau \sim p^\pi(\tau|\mathbf{s}_0=\mathbf{s})}\left[R(\tau)\right],$$

highlighting the anticipated future returns from a specific state $\mathbf{s} \in \mathcal{S}$.

In a similar manner, the *action-value function*, or simply the *Q-function*, denoted as $Q^\pi(\mathbf{s}, \mathbf{a}) : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, expresses the expected return of taking action $\mathbf{a} \in \mathcal{A}$ in state $\mathbf{s} \in \mathcal{S}$ and thereafter following a policy π:

$$Q^\pi(\mathbf{s}, \mathbf{a}) = \mathbb{E}_{\tau \sim p^\pi(\tau|\mathbf{s}_0=\mathbf{s}, \mathbf{a}_0=\mathbf{a})}\left[R(\tau)\right].$$

The Q-function is particularly interesting in the control setting, where it is often necessary to characterize the policy value for not just for states but also for actions.

Both V^π and Q^π can be expressed recursively through the *Bellman equations* as:

$$V^\pi(\mathbf{s}) = \mathop{\mathbb{E}}_{\substack{\mathbf{a}\sim\pi(\mathbf{a}|\mathbf{s}) \\ \mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})}} \left[r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma V^\pi(\mathbf{s}')\right]. \tag{2.15}$$

$$\begin{aligned} Q^\pi(\mathbf{s}, \mathbf{a}) &= \mathop{\mathbb{E}}_{\substack{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a}) \\ \mathbf{a}'\sim\pi(\mathbf{a}'|\mathbf{s}')}} \left[r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma Q^\pi(\mathbf{s}', \mathbf{a}')\right] \\ &= \mathop{\mathbb{E}}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \left[r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma V^\pi(\mathbf{s}')\right]. \end{aligned} \tag{2.16}$$

These recursive definitions provide the basis for many optimization strategies in reinforcement learning. The recursive form of the Q-function can be encoded into a *Bellman operator*, defined as:

$$\begin{aligned} \mathcal{B}^\pi[Q](\mathbf{s}, \mathbf{a}) &\coloneqq \mathop{\mathbb{E}}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \left[r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \mathop{\mathbb{E}}_{\mathbf{a}'\sim\pi(\mathbf{a}'|\mathbf{s}')} \left[Q(\mathbf{s}', \mathbf{a}')\right] \right] \\ &= \mathop{\mathbb{E}}_{\mathbf{s}'\sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \left[r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma V(s')\right] \end{aligned} \tag{2.17}$$

The value functions induce a useful ordering over policies. We say that policy π is better than or equal to π' if its expected return is greater than or equal to that of π' in all states. That is, $\pi \geq \pi' \iff V^{\pi}(\mathbf{s}) \geq V^{\pi'}(\mathbf{s}) \; \forall \; \mathbf{s} \in \mathcal{S}$. Using these, we say that an *optimal policy* π^* is a policy that is better than or equal to all other policies in all states. There always exists at least one, deterministic, optimal policy and all optimal policies share the same optimal state-value function [Puterman, 1994]. We thus denote the optimal state-value function and optimal Q-functions as $V^* \coloneqq V^{\pi^*}$ and $Q^* \coloneqq Q^{\pi^*}$. Furthermore, given the optimal Q-function, an optimal policy can be constructed as: $\pi^*(\mathbf{s}, \mathbf{a}) = \mathbb{1}(\mathbf{a} = \arg\max_{\mathbf{a}'} Q^*(\mathbf{s}, \mathbf{a}'))$, where $\mathbb{1}$ denotes the indicator function.

Analogously to the recursive definitions of V^{π} and Q^{π} in equations (2.15) and (2.16), the optimal state-value function and optimal Q-function are defined recursively through the *Bellman optimality equations* as:

$$V^*(\mathbf{s}) = \max_{\mathbf{a}} Q^*(\mathbf{s}, \mathbf{a}) \tag{2.18}$$

$$= \max_{\mathbf{a}} \mathop{\mathbb{E}}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} [r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma V^*(\mathbf{s}')], \tag{2.19}$$

$$Q^*(\mathbf{s}, \mathbf{a}) = \mathop{\mathbb{E}}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \left[r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \max_{\mathbf{a}'} Q^*(\mathbf{s}', \mathbf{a}') \right]. \tag{2.20}$$

And similarly, the *Bellman optimality operator* for Q-function is defined as:

$$\mathcal{B}^*[Q](\mathbf{s}, \mathbf{a}) \coloneqq \mathop{\mathbb{E}}_{\mathbf{s}' \sim p(\mathbf{s}'|\mathbf{s},\mathbf{a})} \left[r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}') \right] \tag{2.21}$$

For the sake of brevity and where the context unambiguously allows, we will simplify our notation by omitting the superscripts from value functions and Bellman operators. For example, instead of writing V^{π} or $Q^{\pi}(s, a)$, we will use V or $Q(s, a)$, respectively.

Having established the central concepts of agents and environments, we now proceed to introduce methods for solving MDPs, that is obtaining optimal policies π^*.

2.2.4 Planning

In settings where the transition dynamics and reward function of the MDP are fully known, the task of finding optimal control strategies is referred to as *planning*. Planning, though inherently less complex than the broader reinforcement learning setting, which often involves learning these dynamics from interactions with the environment, is nonetheless non-trivial. Effective planning methods systematically explore the space of possible actions and their consequences, leveraging the known model of the environment to predict future states and rewards. This approach enables the anticipation of future scenarios and the formulation of a strategy that maximizes returns.

One of the core concepts in planning is that of *dynamic programming* [Bellman, 1957], a methodological framework that takes into account the recursive or temporal structure of the problem and breaks down complex decision-making problems into smaller, more manageable subproblems. In general, the problems amenable to dynamic programming are required to satisfy two properties: *optimal substructure* and *overlapping subproblems*. In the case of MDPs, optimal substructure is induced through the Markov property giving rise to the recursive decomposition seen in the Bellman equations (equation (2.16)), while the overlapping solutions are a result of the value functions enabling reuse of the solutions.

We begin by introducing two fundamental reinforcement learning algorithms based on dynamic programming: Policy Iteration and Value Iteration.

Policy Iteration

Policy iteration [Howard, 1960] is a method for finding the optimal policy of an MDP by repeatedly alternating between two stages: policy evaluation and policy improvement, which are iteratively applied until convergence to an optimal policy.

The objective of *policy evaluation* is to evaluate the current policy π, i.e. compute the state-value function $Q^\pi(\mathbf{s}, \mathbf{a})$. This evaluation process is typically carried out by iteratively applying the Bellman operator (equation (2.17)) to an arbitrarily chosen initial value function estimate $\hat{Q}_0$ according to the update rule $\hat{Q}_{k+1} = \mathcal{B}[\hat{Q}_k]$.

Under the assumptions that the Bellman operator is a contraction mapping, and the transition dynamics and reward functions meet certain regularity conditions, this iterative process converges to the true Q-function Q^π, reaching a unique fixed point [Sutton and Barto, 2018].

After evaluating the policy π, the *policy improvement* step refines the current policy π to yield a new policy using the Q-function Q^π. The new, improved policy π' is generated to greedily select actions with respect to Q^π as: $\pi'(\mathbf{a}|\mathbf{s}) = \mathbb{1}(\mathbf{a} = \arg\max_{\mathbf{a}'} Q^\pi(\mathbf{s}, \mathbf{a}'))$. The *policy improvement theorem* [Sutton and Barto, 2018] guarantees that the updated policy π' is as good as or better than π for all states and actions.

Through each successive iteration of policy evaluation and improvement, policy iteration converges to the optimal optimal policy π^*. This optimal policy satisfies the Bellman optimality equation equation (2.21) and thereby maximizes expected returns from all states and actions within the MDP.

Generalized policy iteration [Sutton and Barto, 2018] extends this concept by interleaving the policy evaluation and improvement steps more closely, rather than completing each to convergence before proceeding to the next. This approach allows for a more fine-grained update process. As long as both the evaluation and improvement processes are applied to all states, they typically achieve the same outcome: convergence to the optimal value function and an optimal policy.

Value Iteration

Value iteration [Sutton and Barto, 2018] simplifies and combines the policy evaluation and policy improvement steps into a single operation. It directly computes the optimal action-value function, iteratively updating the arbitrarily intialized $\hat{Q}_0$ towards Q^* using the Bellman optimality operator (equation (2.21)): $\hat{Q}_{k+1} = \mathcal{B}^*[\hat{Q}_k]$. After convergence, the optimal policy can again be derived from Q^* exactly as in the policy iteration: $\pi(\mathbf{a}|\mathbf{s}) = \mathbb{1}(\mathbf{a} = \arg\max_{\mathbf{a}'} Q^*(\mathbf{s}, \mathbf{a}'))$.

Value iteration effectively merges the evaluation of Q with its immediate improvement, leading to a more straightforward and often faster convergence to the optimal policy compared to the two-step process of traditional policy iteration.

While both the policy iteration and value iteration have limited practical applicability due to their requirement of full knowledge of the MDP dynamics, they form the foundation for majority of modern model-free reinforcement learning methods that act under unknown environments. We will look into these methods in section 2.2.5.

Next, we will look into methods that can learn optimal policies in environments with unknown dynamics.

2.2.5 Reinforcement Learning

For many of the interesting real-world problems, the environment dynamics — the transition distribution P and the reward function r — are unknown to the agent. Facing decision-making environments with unknown dynamics necessitates *learning* certain aspects of the MDP in order to make educated decisions in them. Reinforcement learning methods use the observed environment transitions $(\mathbf{s}_t, \mathbf{a}_t, r_t, \mathbf{s}_{t+1})$, obtained by interacting with the environment, for learning the optimal policies. The spectrum of reinforcement learning methodologies is broad, each tailored to different problem settings and constraints. While an exhaustive review of these methods is beyond our discussion, we highlight the fundamental categorizations that underpin our work next.

Model-Based vs Model-Free Learning

Reinforcement learning approaches can broadly be divided into two categories based on how they use the environment samples and what they learn from them: model-based and model-free methods.

Model-based methods use the environment samples to explicitly learn approximate models of the transition distribution or the reward function. Once an approximate model $\hat{P} \approx P$ and $\hat{r} \approx r$ are learned, the challenge of finding the

optimal policy simplifies into a planning problem within the simulated dynamics of the learned MDP.

In the ideal scenario, where the learned models accurately represent the real environment, model-based methods are highly sample-efficient. Evaluating the learned models is typically less computationally intensive and faster than obtaining samples from the real environment, making these methods particularly effective for tasks where environmental interactions are costly. Moreover, these methods are often amenable to efficient transfer and the models can potentially be reused across different tasks within the same environment. For example, if the underlying transition model remains valid, only the reward function might need to be adjusted for a new task, facilitating quicker adaptation and learning.

Despite their conceptual simplicity and efficiency, however, model-based methods face a major challenge: distribution shift. This phenomenon arises when the model, accurate within the regions covered by the data that the model is trained on, is used for planning, leading to (virtual) exploration into parts of the state space that are scarcely represented in the training data. The model's predictions become increasingly unreliable in those underrepresented parts of the state space, causing a compounding error through the planning sequence. Each erroneous prediction misguides subsequent decisions, amplifying the inaccuracies.

Learning the models introduces a practical trade-off common in supervised learning. A more expressive model can improve model accuracy at the risk of overfitting to the training data, reducing its usefulness in new situations. This trade-off becomes particularly unforgiving in environments with stochastic or high-dimensional dynamics, as well as with high-capacity function approximators such as neural networks. In the context of real-world applications, when samples from the environment are limited, learning the models requires carefully chosen regularization to avoid overfitting and accurate uncertainty estimates to avoid states where the model is unable to accurately predict the real-world dynamics.

In contrast to model-based methods, *model-free* methods bypass the need to construct an explicit model of the environment dynamics, typically by using the

environment samples to either optimize the reinforcement learning objective directly or to use a policy/value iteration-like process. Since these approaches do not rely on an explicit transition or reward model of the environment, they are less susceptible to errors stemming from model inaccuracies, and are often thought to be easier to implement and tune. Yet they introduce their own set of limitations.

Before we move onto discussing model-free approaches, it is worth noting that the boundary between model-based and model-free methods is loose and many works have combined these approaches, achieving the best of both sides and further blurring the distinction between them. Examples of approaches combining the two include training a model-free agent using data from a learned model [Kalweit and Boedecker, 2017, Lampe and Riedmiller, 2014, Luo et al., 2018, Silver et al., 2008, Sutton, 1990], using models for policy gradient computation [Heess et al., 2015, Nguyen and Widrow, 1990], initializing model-free algorithms from model-based solutions [Farshidian et al., 2014, Levine and Koltun, 2013, Nagabandi et al., 2018], and improving the quality of model-free value functions with model predictions [Buckman et al., 2018, Feinberg et al., 2018].

Value-based vs Policy Gradient Methods

Model-free RL methods can be loosely categorized into *value-based* and *policy gradient* (or policy search) methods. Both approaches aim to optimize the behavior of an agent through interactions with the environment. However, they fundamentally differ in their optimization targets and methodologies.

Value-based RL methods bear close similarity to the value iteration or policy iteration methods discussed in section 2.2.4. The core idea is similar in that if we can accurately estimate the $\hat{Q}^\pi \approx Q^\pi$ for policy π, we can improve the policy such that it greedily maximizes the Q values.

Perhaps the simplest and most intuitive method for estimating the value function is to start with arbitrarily initialized $\hat{Q}^\pi_0$ and iteratively updating this estimate using the Monte Carlo samples from the environment to compute returns. The Monte Carlo estimate for $\hat{Q}^\pi(\mathbf{s}, \mathbf{a}) \approx Q^\pi(\mathbf{s}, \mathbf{a})$ for state-action pair $(\mathbf{s}, \mathbf{a})$ can be obtained

by acting in the environment until the agent reaches the state $\mathbf{s}$, then executing action $\mathbf{a}$, and summing the rewards from there onwards from following the policy π. Concretely, once we encounter the state $\mathbf{s}$ and take the action $\mathbf{a}$, we can compute a sample of the return $R = \gamma^0 r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma^1 r(\mathbf{s}', \mathbf{a}', \mathbf{s}'') + \ldots$ by executing the policy from there onwards. We can then define the error of the current estimate $\hat{Q}_k^\pi(\mathbf{s}, \mathbf{a})$ as:

$$\delta_k = R - \hat{Q}_k^\pi(\mathbf{s}, \mathbf{a}) = (\gamma^0 r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma^1 r(\mathbf{s}', \mathbf{a}', \mathbf{s}'') + \ldots) - \hat{Q}_k^\pi(\mathbf{s}, \mathbf{a}),$$

and use the error to update our value estimate $\hat{Q}^\pi$ for the particular $(\mathbf{s}, \mathbf{a})$-pair using stochastic approximation with learning rate α:

$$\hat{Q}_{k+1}^\pi(\mathbf{s}, \mathbf{a}) = \hat{Q}_k^\pi(\mathbf{s}, \mathbf{a}) + \alpha \delta_k. \tag{2.22}$$

Assuming that the sampling process has a non-zero probability of encountering every state-action pair, in theory, this process eventually allows us to recover the true Q-function Q^π as $k \to \infty$. Unfortunately, in practice this sampling process is prohibitively sample-inefficient for most of the real-world tasks of our interest.

The Bellman operators introduced in equation (2.17) suggest a potential solution to the high-variance problem: instead of estimating the return from the sums rewards of sampled sequences of rewards, we can truncate the summation and substitute the estimate from the value function itself for the truncated tail. This reuse of values, often referred to as *bootstrapping* in RL literature, is one of the most important concepts in reinforcement learning and forms the basis of the so-called *temporal difference*, or TD, methods. Concretely, in this setting, we only observe the next state $\mathbf{s}' \sim p^\pi(\mathbf{s})$, the corresponding reward $r(\mathbf{s}, \mathbf{a}, \mathbf{s}')$, and the next action $\mathbf{a}' \sim \pi(\mathbf{a}'|\mathbf{s})$. Using these, we then define the TD-error as:

$$\delta_k = (r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \hat{Q}_k^\pi(\mathbf{s}', \mathbf{a}')) - \hat{Q}_k^\pi(\mathbf{s}, \mathbf{a}),$$

and update the $\hat{Q}_k^\pi$ according to equation (2.22). This method, commonly referred to as `TD(0)`, can be shown to converge almost surely to the true value of the policy under standard stochastic approximation assumptions.

The temporal difference estimate reduces variance — because its estimates are amortized over multiple samples — with the cost of increased bias — as the bootstrap estimates used may not always be accurate. `TD(0)` and full Monte Carlo estimates present opposite ends of the bias-variance spectrum. Generalizations such as n-step returns [Sutton and Barto, 2018] offer a more nuanced mechanism to control this trade-off. By deferring bootstrapping to the nth step rather than immediately after the first step, one can adjust the bias and variance of the TD approximation. Specifically, as n increases, the approximation becomes less biased but incurs higher variance. Moreover, `TD(`λ`)` [Sutton and Barto, 2018] offers further flexibility by averaging across all possible n-step returns in a weighted manner, with λ determining the weighting scheme. This allows for a more controlled and customizable balance between bias and variance.

Value-based methods have been highly successful in a variety of domains with discrete and reasonably low-dimensional state and action spaces. The appeal of value-based methods lies in their simplicity and directness in learning value functions without requiring explicit policy representations. By iteratively improving the value function approximation, these methods implicitly derive a policy that maximizes expected returns.

However, value-based approaches have limitations, particularly in environments with large or continuous action spaces. In such scenarios, finding the action that maximizes the value function is often intractable, thus limiting their use for practical applications.

Policy gradient methods constitute another fundamental class of model-free reinforcement learning algorithms. Unlike value-based methods that focus on learning a value function to indirectly derive a policy, policy gradient methods directly adjust the policy parameters θ of a parameterized policy π_θ to maximize the expected return.

The core of policy gradient methods is to estimate the gradient $\nabla_\theta J(\theta)$ of the objective function (equation (2.14)) with respect to the policy parameters θ, and employing gradient ascent to update θ. Computing the gradient in its default

form $\nabla_\theta J(\theta) = \nabla_\theta \mathbb{E}_{\tau \sim p^{\pi_\theta}(\tau)}[R(\tau)]$ is difficult, however, as it requires differentiation with respect to the environment dynamics, which are generally unknown to us. To introduce a more practical form of the policy gradient, we first express the policy objective with respect to the policy's marginal state distribution:

$$J(\theta) = \mathop{\mathbb{E}}_{\tau \sim p^{\pi_\theta}(\tau)}[R(\tau)] = \mathop{\mathbb{E}}_{\substack{\mathbf{s} \sim d^{\pi_\theta}(\mathbf{s}) \\ \mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})}}[Q^{\pi_\theta}(\mathbf{s}, \mathbf{a})]$$

where $d^\pi(\mathbf{s})$ denotes the normalized γ-discounted future state occupancy distribution $d_\pi(\mathbf{s}) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t p(\mathbf{s}_t = \mathbf{s}|p_0, p, \pi)$, representing the discounted state distribution induced by the policy π, and $p(\mathbf{s}_t = \mathbf{s}|p_0, p, \pi)$ denotes the likelihood that the agent is in state $\mathbf{s}$ at time step t by following π under the the initial state distribution p_0 and the transition model p. By using the *policy gradient theorem* [Sutton et al., 1999a], the gradient of the objective can be rewritten as:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \mathop{\mathbb{E}}_{\substack{\mathbf{s} \sim d^{\pi_\theta}(\mathbf{s}) \\ \mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})}}[Q^{\pi_\theta}(\mathbf{s}, \mathbf{a})] \\
&= \nabla_\theta \mathop{\mathbb{E}}_{\mathbf{s} \sim d^{\pi_\theta}(\mathbf{s})}\left[\int_{\mathcal{A}} \pi_\theta(\mathbf{a}|\mathbf{s}) Q^{\pi_\theta}(\mathbf{s}, \mathbf{a})\, d\mathbf{a}\right] \\
&= \mathop{\mathbb{E}}_{\mathbf{s} \sim d^{\pi_\theta}(\mathbf{s})}\left[\int_{\mathcal{A}} \nabla_\theta \pi_\theta(\mathbf{a}|\mathbf{s}) Q^{\pi_\theta}(\mathbf{s}, \mathbf{a})\, d\mathbf{a}\right] \\
&= \mathop{\mathbb{E}}_{\mathbf{s} \sim d^{\pi_\theta}(\mathbf{s})}\left[\int_{\mathcal{A}} \pi_\theta(\mathbf{a}|\mathbf{s}) \frac{\nabla_\theta \pi_\theta(\mathbf{a}|\mathbf{s})}{\pi_\theta(\mathbf{a}|\mathbf{s})} Q^{\pi_\theta}(\mathbf{s}, \mathbf{a})\, d\mathbf{a}\right] \\
&= \mathop{\mathbb{E}}_{\mathbf{s} \sim d^{\pi_\theta}(\mathbf{s})}\left[\int_{\mathcal{A}} \pi_\theta(\mathbf{a}|\mathbf{s}) \nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) Q^{\pi_\theta}(\mathbf{s}, \mathbf{a})\, d\mathbf{a}\right] \\
&= \mathop{\mathbb{E}}_{\substack{\mathbf{s} \sim d^{\pi_\theta}(\mathbf{s}) \\ \mathbf{a} \sim \pi_\theta(\mathbf{a}|\mathbf{s})}}[\nabla_\theta \log \pi_\theta(\mathbf{a}|\mathbf{s}) Q^{\pi_\theta}(\mathbf{s}, \mathbf{a})\, d\mathbf{a}]
\end{aligned} \tag{2.23}$$

This forms the theoretical foundation for many different policy gradient algorithms. This formulation still requires samples from the Q-function Q^{π_θ}, which we need to estimate somehow. Methods like `REINFORCE` [Williams, 1992] rely on sample-based Monte Carlo estimate of the returns, which results in a practically applicable, unbiased estimate of the policy gradient. However, as with the value-based methods discussed above, the reliance on sample-based estimates introduces extremely high variance to these methods — affected by both the trajectory length and reward magnitudes — presenting significant practical challenges.

The distinction between value-based and policy gradient methods presents two divergent paths towards achieving reinforcement learning objectives: one through the estimation of value functions and the other by directly optimizing the policy parameters for maximal expected return. However, as alluded to already, both of these approaches come with their own set of limitations. Value-based methods, while effective in discrete action spaces and simpler environments, struggle with the continuous action spaces common in many real-world applications. Policy gradient methods, despite their applicability to both discrete and continuous domains, are often plagued by extremely high variance in their gradient estimates, leading to sample-inefficient learning and instability.

The complementary strengths and weaknesses of value-based and policy gradient methods have led to another class of reinforcement learning algorithms, called *actor-critic* methods, which effectively combine the core ideas of these techniques. By introducing a structure where the policy (the *actor*) is optimized in parallel with a value function estimate (the *critic*), actor-critic methods aim to combine the strengths of both frameworks. The critic aids in stabilizing the training process by providing a baseline against which the policy's performance is evaluated, effectively reducing the variance of the policy gradient estimates without resorting solely to sample-based approaches like `REINFORCE`. This blending addresses the primary challenge of high variance in policy gradient methods while retaining the flexibility to operate across a broad spectrum of environments and action spaces.

The iterative refinement of the actor guided by the critic mirrors the principles of generalized policy iteration discussed earlier in section 2.2.4, bridging the conceptual gap between classical dynamic programming methods and modern reinforcement learning algorithms. The iterative process allows for a more flexible, stable, and sample-efficient policy training, and many of the more widely-adopted reinforcement learning algorithms nowadays follow the actor-critic structure.

Exploration-Exploitation Trade-Off

The exploration-exploitation trade-off is a central challenge in RL, crucial to the efficiency of the learning process in environments where the dynamics are initially unknown to the agent. This trade-off forces the agent to continually decide between two strategies: exploring new actions to better understand the environment and exploiting its current knowledge to maximize rewards. The right balance between these two strategies is crucial for the development of an effective learning algorithm.

Exploration refers to the strategy of making choices that may not have immediate benefits but are valuable for gaining information. This information-seeking behavior enables the agent to learn about the environment, including the short-term and long-term consequences of its actions. While exploratory actions can be seen as sub-optimal and costly in the short term, their long-term consequences may enable more informed decisions in the future and thus outweigh their short-term cost.

Exploitation, on the other hand, refers to choosing actions that the agent currently believes to offer the highest expected return. This approach leverages the agent's accumulated knowledge to make decisions that are expected to yield the most immediate reward. The challenge with a purely exploitative strategy is that it can lead the agent to get stuck in local optima, particularly if the agent's understanding of the environment is incomplete or inaccurate. Thus, while exploitation is essential for realizing the benefits of learning, it must be balanced with exploration to ensure that the agent does not overlook potentially superior strategies.

The dynamic between exploration and exploitation is influenced by the stage of learning. Early on in the training, when the agent has limited knowledge of the environment, it may benefit more from exploration to accumulate valuable information. As the agent learns more about its environment, the emphasis can shift more towards exploitation, making use of the knowledge gained to achieve higher rewards.

Exploration strategies can be implemented in various ways. The simplest exploration methods omit explicit exploration altogether and rely on, for example, the *residual stochasticity* in the agent's policy. For example, Gaussian policies

will introduce noise to the actions until the policy is fully optimized, thus leading to explorative actions. In a similar fashion, stochastic approximation methods introduce noise to the optimization, which might lead to exploration.

Alternatively, some techniques add explicit noise to the agent's policy. ϵ-*greedy* strategy, for example, selects the agent's estimated optimal action with probability $1 - \epsilon$ and a uniform random action with probability ϵ. *Boltzmann-exploration*, on the other hand, chooses actions relative to the estimated Q-function values. Such methods are often referred to as *dithering exploration* methods. While computationally efficient and easy to implement, dithering exploration lacks a directed strategy based on the agent's understanding of its environment. This limitation can lead to inefficient exploration, where the agent spends time exploring areas of the environment that offer little potential for high returns. Despite their suboptimal behavior, these methods are some of the most widely-used exploration approaches.

In contrast, uncertainty-driven exploration seeks to adapt its strategy based on the agent's evolving uncertainty in the MDP. As the agent learns more about its environment, its epistemic uncertainty naturally decreases. This allows it to confidently avoid actions likely to lead to low returns and instead focus its exploration on areas that still hold the potential for high rewards. Bayes-optimal exploration provides the gold standard for uncertainty-driven exploration, optimally trading off exploration and exploitation. Computing such policies is often intractable, however, and various approximations are used in practice. Perhaps the most common of these are posterior sampling [Osband et al., 2013, Strens, 2000, Thompson, 1933] and optimism-based [Ciosek et al., 2019, Kaelbling, 1994, Kolter and Ng, 2009, Wang et al., 2005] methods. We come back to the uncertainty-driven exploration in chapter 3.

It is worth differentiating between *myopic* and *deep* exploration strategies [Osband et al., 2016, 2019a]. Myopic exploration focuses on immediate uncertainties, often considering only the near-term future outcomes of actions. Deep exploration, conversely, looks at the long-term consequences, including how current choices can

uncover information that affects future decisions and rewards. This distinction is crucial for understanding how an agent can strategically navigate its learning process to optimize for both immediate and future gains. Deep exploration is particularly relevant in complex environments with long-term dependencies and where understanding the full scope of possible outcomes is essential for optimal performance.

The recent surveys by [Amin et al., 2021, Hao et al., 2023, Yang et al., 2021] provide a more comprehensive overview on exploration in RL.

2.2.6 Problem Settings

So far in our discussion we have assumed the standard MDP formulation introduced in section 2.2.1. While this framework offers a flexible model for sequential decision making, it does not capture the full spectrum of nuances and specific challenges present in a variety of real-world problem settings. Recognizing this, it is often desirable to extend or constrain the standard MDP formulation to better align with the particular inductive biases or the unique characteristics of the problem setting and the anticipated solutions. We will next briefly discuss some of the MDP extensions relevant to our investigation.

Partial Observability

In real-world applications, such as robotics, agents frequently encounter noisy or incomplete sensor readings, leading to *partial observability* — that is, the sensor observation received by the agent is insufficient to infer the future state of the system — thereby breaking the Markovian assumption inherent in the standard MDP formulation. To accommodate partial observability, the MDP framework can be extended to the Partially observable Markov decision process (POMDP) framework [Kaelbling et al., 1998]. A POMDP is defined by a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, O, P, P_0, r, \gamma)$, where the $\mathcal{O}$ represents the observation space, and O denotes the observation distribution, described by the probability density $o(\mathbf{o}|\mathbf{a}, \mathbf{s}) : \mathcal{O} \times \mathcal{A} \times \mathcal{S} \to [0, \infty)$ of receiving

observation $\mathbf{o}$ given state $\mathbf{s}$ and action $\mathbf{a}$. The rest of the components of the tuple are as defined for an MDP.

When the observations are non-Markovian, the agent must act under uncertainty of the true environment state and the optimal behaviors may necessitate actions that are taken purely because they improve the agent's estimate of the current state, thereby enabling better future decisions. The agent is commonly conditioned on the entire history of observations and actions $(o_{\leq t}, a_{<t})$. In this work, the decision making tasks considered in chapter 4 incorporate partial observability.

Transfer and Multi-Task Learning

In the standard reinforcement learning setting, the agent is typically tasked to master a single task within a specific MDP. This approach, while effective for isolated problems, does not exploit the potential benefits of knowledge transfer across different but related tasks — a common theme in machine learning in general, where leveraging shared insights can significantly enhance learning outcomes. Recognizing this limitation, the concepts of *multi-task* and *transfer* in reinforcement learning have emerged to enhance learning efficiency and effectiveness by leveraging shared information across multiple domains or tasks.

Multi-task learning involves training a single agent across multiple MDPs, or tasks, that are sampled from a predefined yet unknown task distribution. This method operates under the assumption that the skills, strategies, or knowledge required to solve one task may have relevance to others within the same distribution. By learning several tasks in parallel, the model can develop shared representations and strategies, thereby fostering a more generalized, robust, and sample-efficient learning mechanism. The key objective is to improve performance not only on individual tasks by exploiting the commonalities and differences across tasks but also to enhance overall learning efficiency and the transferability of acquired skills. Formally, the multi-task extension of the standard MDP framework introduces a set of tasks, where each task, identified with $k \in \mathcal{K}$, is characterized by its own MDP $(\mathcal{S}, \mathcal{A}, P^k, P_0^k, r^k, \gamma)$ but shares the state and action spaces $(\mathcal{S}, \mathcal{A})$ across the task

set. The environment dynamics (P^k, P_0^k, r^k) can vary from task to task, although a common structural foundation is assumed.

Transfer learning, in contrast, focuses on the sequential application of knowledge gained from one or more source tasks to improve or accelerate learning in a target task. It is especially relevant in scenarios where acquiring knowledge from scratch for every new task is impractical due to resource constraints or when the target domain suffers from limited data availability. Transfer learning involves strategies for identifying relevant knowledge from source domains $\mathcal{K}_{\text{source}}$ and applying it to a target domain $\mathcal{K}_{\text{target}}$, thereby addressing challenges related to learning efficiency, scalability, and generalization across domains. Unlike multi-task learning, where tasks are learned simultaneously and sampled from a common distribution, transfer learning typically involves a more distinct sequence of tasks, leading to potentially abrupt shifts in domain or task distribution. This distinction highlights the adaptive nature of transfer learning, as it seeks to tailor and apply previously acquired knowledge to new and often significantly different tasks.

Motor Control

The versatility of the MDP framework allows it to be applied across a vast array of problems, each with its unique challenges and characteristics. In the realm of decision-making, the temporal aspect represented by time steps in an MDP need not correspond to fixed intervals of real time but rather to successive stages of decision-making and action execution. These actions could range from low-level controls, such as adjusting the voltages to the motors of a robot arm, to high-level strategic decisions, like choosing a career path or deciding when to eat. Similarly, the concept of states within an MDP is remarkably flexible. States can be defined by direct sensor inputs, embodying low-level perceptions, or they might encapsulate more abstract, high-level information, such as symbolic descriptions of the environment. Moreover, states can incorporate memory of past events or even purely mental states, like uncertainty or surprise, thereby broadening the framework's applicability beyond physical to computational and cognitive domains.

This thesis mainly focuses on the domain of motor control within physically-simulated environments, characterized by continuous actions and state spaces. These environments often simulate embodied systems, such as robots or virtual characters, governed by the laws of physics (or their numerical approximations). Specifically, we consider agents modeled as a hierarchy of articulated rigid bodies, where each body segment, or "body", is connected to another through joints that may be actuated by motors. This arrangement enables complex, articulated movements akin to those observed in natural organisms.

In this context, the state of an agent at any given time, denoted as $\mathbf{s} = [\mathbf{q}, \mathbf{v}, \mathbf{s_a}]$, encapsulates the positions $\mathbf{q}$ and velocities $\mathbf{v}$ of the joints, along with the actuator states $\mathbf{s_a}$, which could be null if the model does not involve stateful actuators. The agent influences its environment through actions $\mathbf{a}$, which are translated into motor torques applied at the joints, facilitating movement and interaction with the environment.

Throughout this thesis, we explore several instances of such motor-controlled systems, from the simple dynamics of a Cartpole in section 3.6.2, through the manipulation with Robotic Arm in figure 4.2, to the more complex behaviors of a full-body humanoid characters in figure 5.2.

3

Bayesian Bellman Operators

Contents

This chapter is based on the Bayesian Bellman Operators work [Fellows, Hartikainen, and Whiteson, 2021] and was done in collaboration with Mattie Fellows. The theory is largely due to Mattie.

3.1 Introduction

In the previous chapters, we highlighted the fundamental exploration-exploitation trade-off in reinforcement learning. We begin by investigating methods to directly improve exploration within model-free reinforcement learning algorithms.

A Bayesian approach to reinforcement learning (RL) characterizes uncertainty in the Markov decision process (MDP) via a posterior [Ghavamzadeh et al., 2016, Vlassis et al., 2012]. A great advantage of Bayesian RL is that it offers a natural and elegant solution to the exploration/exploitation problem, allowing the agent to explore to reduce epistemic uncertainty in the MDP, but only to the extent that exploratory actions lead to greater expected return; unlike in heuristic strategies such as ε-greedy and Boltzmann sampling, the agent does not waste samples trying actions that it has already established are suboptimal, leading to greater sampling efficiency. Elementary decision theory shows that the only admissible decision rules are Bayesian [Cox and Hinkley, 1974] because a non-Bayesian decision can always be Pareto improved upon by a Bayesian agent [de Finetti, 1937]. In addition, pre-existing domain knowledge can be formally incorporated by specifying priors.

In model-free Bayesian RL, a posterior is inferred over the Q-function by treating samples from the MDP as stationary labels for Bayesian regression. A major theoretical issue with existing model-free Bayesian RL approaches is their reliance on bootstrapping using a Q-function approximator, as samples from the exact Q-function are impractical to obtain. This introduces error as the samples are no longer estimates of a Q-function and their dependence on the approximation is not accounted for.

This chapter introduces *B*ayesian *B*ellman *O*perators (BBO), a novel model-free Bayesian RL framework that aims to address such limitations of existing model-free Bayesian methods. Our focus is on the practical application and empirical validation of this framework. We showcase how the BBO can be used to derive gradient-based algorithms for Bayesian policy evaluation and uncertainty estimation, emphasizing the applicability and effectiveness of these methods through comprehensive experimental analyses. Notably, in the policy evaluation setting,

our experiments provide evidence for the consistency and convergence properties of the BBO framework under conditions of nonlinear function approximators and approximate inference.

In the control setting, we derive a Bayesian actor-critic algorithm, forming a continuous counterpart to BootDQN+Prior by Osband et al. [2018]. The lagged target parameters, which are essential to BootDQN+Prior and to the performance of many other model-free RL algorithms, arise cleanly from applying approximate inference to the BBO posterior. These lagged target parameters cannot be explained by existing model-free Bayesian RL theory. Our experimental analysis also provides evidence for their necessity in practice. Furthermore, our algorithm can learn optimal policies in domains where state-of-the-art actor-critic algorithms like Soft Actor-Critic (SAC) [Haarnoja et al., 2018b] fail catastrophically due to their inability to properly explore.

3.2 Bayesian Reinforcement Learning

3.2.1 Bayes-Adaptive Markov Decision Process

In reinforcement learning, our goal is to find an optimal behavioral policy that maximizes expected return according to the objective in equation (2.12). Evaluating this objective assumes that the environment's transition distribution P and reward function r are known. Often, we do not have perfect knowledge of P or r in advance, rendering the objective impossible to evaluate directly. This mismatch presents a challenge. Since the standard frequentist RL objective doesn't model uncertainty in the environment, an agent relying on it faces an inescapable exploration-exploitation dilemma. It must constantly balance the need to learn more about P and r through exploration against the desire to act greedily, maximizing rewards based on its current knowledge.

This limitation motivates the Bayesian view for RL. A Bayes-adaptive Markov decision process (BAMDP) is a framework that addresses the exploration-exploitation dilemma by integrating uncertainty about the environment directly into the decision-making process [Duff and Barto, 2002, Ghavamzadeh et al., 2016]. Unlike a

traditional MDP, where the transition dynamics and reward functions are assumed to be known and fixed, a BAMDP models the environment's uncertainty through a posterior which characterizes the agent's belief over the possible MDP. This belief distribution is updated as the agent interacts with the environment, gathering more evidence about the true nature of the MDP. Each action not only affects the immediate reward and the next state but also updates the agent's belief about the environment, essentially treating the learning process as part of the environment itself, thereby making the exploration-exploitation trade-off an inherent aspect of decision-making in BAMDPs.

3.2.2 Model-based Bayesian RL

BAMDP-optimal policies are the gold standard, optimally balancing exploration with exploitation, but require maintaining a posterior over the unknown transition distribution which is typically challenging to compute due to its high-dimensionality and multi-modality [Song et al., 2013]. Furthermore, planning in BAMDPs requires the calculation of high-dimensional integrals that render the problem hopelessly intractable for all but the simplest problems. In addition, computing the Bayes-optimal policy requires considering every possible history that the agent might encounter. Even with approximation, most existing methods are restricted to small and discrete state-action spaces [Asmuth and Littman, 2011, Guez et al., 2013]. Some attempts have been made [Yao et al., 2021, Zintgraf et al., 2020, 2021] to make Bayesian model-based RL more tractable by being Bayesian over a limited set of model parameters. While these methods are not fully Bayes-optimal, they offer a compromise by introducing parametric models of reward and state transitions, then inferring a posterior over a smaller, more manageable random variable. This keeps computations feasible while still incorporating some elements of Bayesian uncertainty.

As noted by Fellows [2021], in high-dimensional problems, accurately modeling the state transition distribution in a model-based Bayesian RL setting becomes

computationally intractable. This arises from two main challenges. First, representing a complex, high-dimensional transition distribution requires an extensive number of parameters in the model. The large number of parameters translates into a significant computational burden when performing inference, which involves calculations that integrate over the entire parameter space. Second, learning these distributions accurately requires a massive amount of data. As the dimensionality of the state space increases, the number of samples needed to achieve a desired level of accuracy grows exponentially due to the curse of dimensionality. One would also expect the posterior distribution over the model parameters to concentrate at a slower rate in higher dimensions, meaning that the agent would require even more data to reduce its uncertainty about the true transition dynamics.

3.2.3 Posterior Sampling and Optimism

Given the computational challenges of fully Bayesian model-based RL, we next discuss two alternative strategies that leverage posterior information more tractably. While these approaches do not achieve strict Bayes-optimality, they mitigate the computational overhead and remain practical for larger and more complex problems.

One such approach is posterior sampling [Osband et al., 2013, Strens, 2000], which extends Thompson sampling [Thompson, 1933] from simpler bandit problems to MDPs. In posterior sampling, the agent periodically draws a single sample MDP $\phi \in \Phi$ from its posterior belief about the environment. It then computes the optimal policy $\pi^\star_\phi \in \arg\max_\pi J(\pi|\phi)$ for that MDP ϕ and follows that policy for a certain period. As the agent gathers more data from the environment, its posterior P_Φ over MDPs concentrates, which subsequently narrows down the exploration.

Another strategy is based on the principle of *optimism in the face of uncertainty* (OFU) [Ciosek et al., 2019, Kaelbling, 1994, Kolter and Ng, 2009, Wang et al., 2005]. OFU encourages exploration by deliberately overestimating the potential of uncertain actions. The method constructs a set of optimistic MDP parameterizations $\Phi^{\text{optimistic}}$. Each parameterization within $\Phi^{\text{optimistic}}$ is chosen because its expected optimistic returns, under its corresponding optimal policy $\pi^\star_\phi$, rank highly among all

considered MDPs Φ. The agent then follows the optimal policy $\pi^\star_\phi \in \arg\max_\pi J(\pi|\phi)$ for a randomly-chosen optimistic MDP $\phi \in \Phi^{\text{optimistic}}$, which either results in high-value actions or learning from experience where the optimism was unwarranted. In practice, OFU involves maintaining a posterior and upper confidence bound over an optimal Q-function. Actions are then selected using an exploration bonus that combines the predictive Q-function with an additional term that adds an optimistic value proportional to the state-action visitation frequencies of the agent.

Both posterior sampling and OFU provide tractable means to approximate Bayesian exploration, with both achieving near-optimal Bayesian regret bounds [Osband and Van Roy, 2017, Osband et al., 2013]. However, as discussed by Osband and Van Roy [2017], posterior sampling often enjoys a computational advantage as it only requires sampling a model once per episode and then applying standard planning techniques. In contrast, OFU may involve solving complex optimization problems for the upper confidence bounds. These bounds can be challenging to construct, particularly with large or continuous spaces. As supported by both theoretical and experimental findings by [Osband and Van Roy, 2017], posterior sampling often proves to be the more robust method in practice. Therefore, we will focus on posterior sampling when introducing practical algorithms within this chapter.

3.2.4 Model-free Bayesian RL

Existing model-free Bayesian RL approaches attempt to solve a Bayesian regression problem to infer a posterior predictive over a value function [Ghavamzadeh et al., 2016, Vlassis et al., 2012]. While forgoing the ability to separately model reward uncertainty and transition dynamics, modeling uncertainty in a value function avoids the difficulty of estimating high-dimensional conditional distributions and mimics a low-dimensional Bayesian regression problem, for which there are tractable approximate methods [Beal, 2003, Gal, 2016, Jordan, 1999, Kingma and Welling, 2014, Lakshminarayanan et al., 2017, Rezende and Mohamed, 2015]. These methods assume access to a dataset of N samples: $\mathcal{D}^N \coloneqq \{q_i\}_{i=1}^N$ from a distribution over the true Q-function at each state-action pair: $q_i \sim P_Q(\cdot|s_i, a_i)$. Each sample is an

estimate of a point on the true Q-function $q_i = Q^\pi(s_i, a_i) + \eta_i$ corrupted by noise η_i. By introducing a probabilistic model of this random process, the posterior over the Q-function $P(Q^\pi|s, a, \mathcal{D}^N)$ can be inferred, which characterizes the aleatoric uncertainty in the sample noise and epistemic uncertainty in the model. Modeling aleatoric uncertainty is the goal of distributional RL [Bellemare et al., 2017]. In Bayesian RL we are more concerned with epistemic uncertainty, which can be reduced by exploration [Osband et al., 2018].

3.2.5 Issues with Existing Model-free Approaches

Unfortunately, for most settings it is impractical to sample directly from the true Q-function. To obtain efficient algorithms, the samples q_i are approximated using bootstrapping: here a parametric function approximator $\hat{Q}_\omega : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ parameterized by $\omega \in \Omega$ is learned as an approximation of the Q-function $\hat{Q}_\omega \approx Q^\pi$ and then a TD sample is used in place of q_i. For example, a one-step TD estimate approximates the samples as: $q_i \approx r_i + \gamma \hat{Q}_\omega(s_i, a_i)$, introducing an error that is dependent on ω. Existing approaches do not account for this error's dependency on the function approximator. When bootstrapping, the samples are no longer noisy estimates of a point $Q^\pi(s_i, a_i)$ and the resulting posterior is not $P_Q(\cdot|s, a, \mathcal{D})$ as it has dependence on $\hat{Q}_\omega$ due to the dataset. This problem is made worse when a posterior is inferred over an optimal Q-function as it is impossible to sample from the optimal policy a priori to obtain unbiased samples.

In this chapter, we present a model-free Bayesian RL framework, *B*ayesian *B*ellman *O*perators (BBO), designed to address the aforementioned issues. Our focus is on the practical application of the BBO framework and we derive approximate inference algorithms for both policy evaluation and control under nonlinear function approximators, for which the posterior normalization is intractable in general. Through a series of practical experiments, we demonstrate that the algorithms derived from the BBO framework exhibit sophisticated deep exploration properties that enable them to solve continuous control tasks at which state-of-the-art regularized actor-critic algorithms fail catastrophically. Additionally, our

experiments provide evidence for the algorithms' consistency with frequentist RL solutions and their convergence even under nonlinear function approximation where many TD-based methods are known to diverge.

3.3 Bayesian Bellman Operators

As motivated above, directly modeling uncertainty in high-dimensional transition and reward distributions often proves intractable. Instead, modeling uncertainty within a lower-dimensional space, such as the value function or Bellman operator, is more desirable. Model-free Bayesian RL enables this approach. Furthermore, as discussed in section 2.2.5, we want to incorporate bootstrapping to improve the algorithm's sample efficiency and reduce variance.

Considering the aforementioned limitations of existing Bayesian RL approaches, our goal in this section is to introduce a method that fulfills the following criteria: 1) Employs a model-free approach to reinforcement learning and enables the use of bootstrapped samples. 2) Effectively characterizes epistemic uncertainty within the MDP, promoting deep and adaptive exploration strategies. 3) Is applicable in continuous control settings.

3.3.1 Uncertainty in the Bellman Operator

To address the problem of finding a Q-function in a BAMDP, this section introduces the *B*ayesian *B*ellman *O*perators (BBO) framework. We define the Bayesian Bellman operator as:

$$\mathcal{B}\left[Q^\pi\right](s,a,\mathcal{D}) = \mathbb{E}_{P^\pi(r,s',a'|s,a,\mathcal{D})}\left[r + \gamma Q^\pi(s',a',\mathcal{D})\right],$$

with $P^\pi(r,s',a'|s,a,\mathcal{D}) = \pi(a'|s')\,P(r|s,a,s',\mathcal{D})\,P(s'|s,a,\mathcal{D})$. Similar to the frequentist Bellman equations, any Bayesian Q-function Q^π satisfies a Bayesian Bellman equation $\mathcal{B}\left[Q^\pi\right](\cdot,\mathcal{D}) = Q^\pi(\cdot,\mathcal{D})$.

In order to learn a Bayesian Q-function $Q^\pi(\cdot,\mathcal{D})$, we introduce a parametric function approximator $\hat{Q}_\omega \approx Q^\pi(\cdot,\mathcal{D})$ with parameters ω. Our objective is to find $\omega^\star$ such that $\hat{Q}_{\omega^\star} = \mathcal{B}\left[Q^\pi\right](\cdot,\mathcal{D})$. A simple approach to learning such $\omega^\star$ would

be to minimize the mean squared Bayesian Bellman error (MSBBE) between the Bellman operator and the function approximator:

$$\text{MSBBE}(\omega, \mathcal{D}) \coloneqq \left\| \mathcal{B}[\hat{Q}_\omega](\cdot, \mathcal{D}) - \hat{Q}_\omega \right\|^2_{\rho,\pi}, \tag{3.1}$$

where the distribution on the ℓ_2-norm has support over states and actions. Evaluating $\mathcal{B}[\hat{Q}_\omega](\cdot, \mathcal{D})$, while theoretically possible in the Bayesian setting, is intractable due to the integrals over reward and transition distributions. Instead, we introduce a change of variables using the empirical Bellman function, b, defined as:

$$b = b^\pi_\omega(r, s') \coloneqq r + \gamma \, \mathbb{E}_{\pi(a'|s')} \left[\hat{Q}_\omega(s', a') \right]. \tag{3.2}$$

The transformed variable $B : \mathcal{R} \to \mathcal{R}$ has a conditional distribution $P_B(\cdot|s, a, \mathcal{D}_\omega)$, which is the pushforward of $P^\pi(s', a'|s, a)$ under the transformation b and satisfies:

$$\begin{aligned}
\mathcal{B}\left[\hat{Q}_\omega\right](s, a, \mathcal{D}) &= \mathop{\mathbb{E}}_{P^\pi(r,s',a'|s,a,\mathcal{D})} \left[r + \gamma \hat{Q}_\omega(s', a') \right] \\
&= \mathop{\mathbb{E}}_{P(r,s'|s,a,\mathcal{D})} \left[r + \gamma \, \mathbb{E}_{\pi(a'|s')} \left[\hat{Q}_\omega(s', a') \right] \right] \\
&= \mathop{\mathbb{E}}_{P(r,s'|s,a,\mathcal{D})} \left[b^\pi_\omega(r, s') \right] \\
&= \mathop{\mathbb{E}}_{P_B(b|s,a,\mathcal{D}_\omega)} [b]
\end{aligned}$$

The pushforward $P_B(\cdot|s, a, \mathcal{D}_\omega)$ is a distribution over empirical Bellman functions, characterizing the uncertainty in the true Bellman operator $\mathcal{B}[\hat{Q}_\omega]$ of the function approximator $\hat{Q}_\omega$.

As discussed in section 3.2.4, modeling the uncertainty in the scalar-valued Bellman operator is much simpler than in the transition distribution P or reward function r [Fellows, 2021].

3.3.2 BBO Model

Our goal is to infer the posterior $P_B(b|s, a, \omega)$ over Bellman operators. The graphical model for BBO is shown in figure 3.1. To model $P_B(b|s, a, \omega)$ we assume a parametric conditional distribution $P(b|s, a, \phi)$ with model parameters $\phi \in \Phi$ and likelihood $p(b|s, a, \phi)$. We write the likelihood's conditional mean as $\hat{B}_\phi(s, a) \coloneqq \mathbb{E}_{P(b|s,a,\phi)}[b]$, which can be interpreted as a function space of approximators that represents a

space of Bellman operators, each indexed by $\phi \in \Phi$. The choice of $P(b|s, a, \phi)$ should therefore ensure that the space of approximate Bellman operators characterized by $\hat{B}_\phi$ is expressive enough to sufficiently represent the true Bellman operator. We also represent our pre-existing beliefs in the true Bellman operator by specifying a prior P_Φ.

We collect a dataset $\mathcal{D}_\omega^N \coloneqq \{(s_i, a_i, b_i)\}_{i=1}^N$ consisting of N samples from the environment, using a standard model-free RL procedure where the agent interacts with in the environment to generate environment transitions. Specifically, starting from state s_i, the agent selects an action $a_i \sim \pi(\cdot|s_i)$, transitions to the next state $s_i' \sim P(\cdot|s_i, a_i)$, and receives a reward $r_i = r(s_i, a_i, s_i')$. To obtain a sample b_i of the random variable b, we apply the empirical Bellman function, $b_i = r_i + \gamma\, \mathbb{E}_{\pi(a_i'|s_i')}\left[\hat{Q}_\omega(s_i', a_i')\right]$ as in equation (3.2), to the observed transitions. In practice, the expectation within the Bellman function is approximated by sampling actions from the policy, yielding unbiased samples b_i of the Bellman operator. As we discussed in section 3.2.5, existing model-free Bayesian RL approaches incorrectly treat each b_i as a sample from a distribution over the value function $P_Q(Q^\pi|s, a)$. BBO corrects this by modeling the true conditional distribution: $P_B(b|s, a, \omega)$ that generates the data.

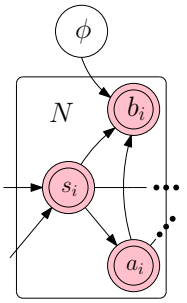


Figure 3.1: Graphical Model for BBO.

As we are not concerned with modeling the transition distribution in our model-free paradigm, we assume states are sampled either from an ergodic Markov chain or i.i.d. from a replay buffer. Off-policy samples can be corrected using importance sampling.

Assumption 3.3.1 (State Generating Distribution)**.** Each state s_i is drawn either i) i.i.d. from a distribution $\rho(s)$ with support over $\mathcal{S}$ or ii) from an ergodic Markov chain with stationary distribution $\rho(s)$.

Given the prior P_Φ and a dataset $\mathcal{D}_\omega^N$ of samples from the true distribution P_B, using Bayes' rule, we can infer the posterior over parameters $P_\Phi(\phi|\mathcal{D}_\omega^N)$, which

has the density (see appendix A.1.1 for a derivation using both state generating distributions of assumption 3.3.1):

$$p_\Phi(\phi|\mathcal{D}_\omega^N) = \frac{\prod_{i=1}^N p(b_i|s_i,a_i,\phi)p_\Phi(\phi)}{\int_\Phi \prod_{i=1}^N p(b_i|s_i,a_i,\phi)\,dP_\Phi(\phi)}. \tag{3.3}$$

To be able to make predictions, we infer the posterior predictive over Bellman operators $p_B(b|s,a,\mathcal{D}_\omega^N)$ by marginalizing over the parameters in the likelihood: $p_B(b|s,a,\mathcal{D}_\omega^N) = \int_\Phi p(b|s,a,\phi)\,dP(\phi|\mathcal{D}_\omega^N)$. Unlike existing approaches, this posterior density is a function of ω, which correctly accounts for the dependence on $\hat{Q}_\omega$ in our data and the generating distribution $P_B(b|s,a,\mathcal{D}_\omega^N)$.

We highlight that it is possible to define a likelihood and prior that are functions of ω to encode any prior knowledge of how the underlying Bellman operator varies with ω, however this is not strictly necessary as the posterior automatically accounts for this dependence due to its conditioning on $\mathcal{D}_\omega^N$. As we anticipate that most applications of BBO will seek to learn an optimal policy, and hence an optimal Q-function, a prior P_Φ that incorporates any knowledge available about the optimal Bellman operator will speed learning and give the agent an advantage.

As our data depends on $\hat{Q}_\omega$, we must introduce a method of learning the correct Q-function approximator. As every Bellman operator characterizes an MDP, the posterior predictive mean represents a Bayesian estimate of the true MDP by using the posterior to marginalize over all Bellman operators that our model can represent according to our uncertainty in their value:

$$\mathcal{B}_{\omega,N}(s,a) \coloneqq \mathbb{E}_{P_B(b|s,a,\mathcal{D}_\omega^N)}[b] = \mathbb{E}_{P_\Phi(\phi|\mathcal{D}_\omega^N)}\left[\hat{B}_\phi(s,a)\right]. \tag{3.4}$$

The predictive mean $\mathcal{B}_{\omega,N}$ is the Bayesian Bellman operator under our BBO model and our Q-function approximator should satisfy a Bellman equation using $\mathcal{B}_{\omega,N}$. As a simple approach to learn $\omega^\star$ is to minimize the mean squared Bayesian Bellman error (MSBBE) from equation (3.1) between the posterior predictive and function approximator:

$$\text{MSBBE}_N(\omega) \coloneqq \left\|\hat{Q}_\omega - \mathcal{B}_{\omega,N}\right\|_{\rho,\pi}^2 \tag{3.5}$$

Although the MSBBE has a similar form to a mean squared Bellman error with a Bayesian Bellman operator in place of the Bellman operator, Fellows et al. [2021] show that its frequentist interpretation is closer to the mean squared projected Bellman error (MSPBE) [Sutton et al., 2009b] used by convergent frequentist TD algorithms. The gradient of the MSBBE with respect to the function approximator parameters ω is (see appendix A.1.2 for derivation):

$$\begin{aligned}
&\nabla_\omega \text{MSBBE}_N(\omega) \\
&\propto \mathbb{E}_{\rho,\pi}\left[\left(\hat{Q}_\omega - \mathbb{E}_{P_\Phi(\phi|\mathcal{D}_\omega^N)}\left[\hat{B}_\phi\right]\right)\left(\nabla_\omega \hat{Q}_\omega - \nabla_\omega \mathcal{B}_{\omega,N}\right)\right] \\
&= \mathbb{E}_{\rho,\pi}\left[\left(\hat{Q}_\omega - \mathbb{E}_{P_\Phi(\phi|\mathcal{D}_\omega^N)}\left[\hat{B}_\phi\right]\right)\left(\nabla_\omega \hat{Q}_\omega - \mathbb{E}_{P_\Phi(\phi|\mathcal{D}_\omega^N)}\left[\hat{B}_\phi \nabla_\omega \log p_\Phi(\phi|\mathcal{D}_\omega^N)\right]\right)\right]. \quad (3.6)
\end{aligned}$$

If we can sample from the posterior, then unbiased estimates of $\nabla_\omega \text{MSBBE}_N(\omega)$ can be obtained using equation (3.6), hence minimizing the MSBBE via a stochastic gradient descent algorithm is convergent if the standard Robbins-Munro conditions are satisfied [Robbins and Monro, 1951]. Minimizing the MSBBE also avoids the double-sampling problem encountered in frequentist RL, where, to minimize the mean squared Bellman error, two independent samples from $P(s'|s,a)$ are required to obtain unbiased gradient estimates [Baird, 1995], which is not possible because $P(\cdot|s,a)$ is unknown to the agent a priori. In BBO, this issue is avoided by drawing two independent approximate Bellman operators $\hat{B}_{\phi_1}$ and $\hat{B}_{\phi_2}$ from the posterior $\phi_1, \phi_2 \sim P_\Phi(\cdot|\mathcal{D}_\omega^N)$ instead.

Many existing model-free Bayesian RL approaches attempt to carry out policy evaluation by minimizing objectives similar to the MSBBE but using posterior over Q-functions instead of the BBO posterior. As their posterior, $P_Q(s,a,\mathcal{D})$, has no dependence on ω, the gradient $\nabla_\omega \log p_\Phi(\phi|\mathcal{D}_\omega^N)$ is not accounted for, leading to gradient terms being dropped in the update. Stochastic gradient descent using these updates does not optimize any objective and so may not converge to any solution. Within the Bayesian regime, this issue extends to impacting exploration quality. Specifically, the lack of convergence may prevent the posterior $P_B(\cdot|s,a,\mathcal{D}_\omega^N)$ from concentrating and reduces the posterior's ability to correctly characterize the uncertainty in the Bellman operator, undermining the benefits

of uncertainty-based exploration. This is concretely evidenced in our empirical evaluations in figure 3.9 in section 3.6.2.

Fellows et al. [2021] prove that under mild regularity assumptions, the posterior concentrates on the frequentist solution $\phi^\star_\omega$ and thus the Bayesian Bellman operator converges to the true Bellman operator $\hat{B}_{\phi^\star_\omega}(s,a) = \mathbb{E}_{P(b|s,a,\phi^\star_\omega)}[b] = \mathbb{E}_{P(b|s,a,\omega)}[b] = \mathcal{B}[\hat{Q}_\omega](s,a)$ almost surely. As every Bellman operator characterizes an MDP, any Bayesian RL solution obtained using the BBO posterior, such as an optimal policy or value function, is consistent with the true RL solution. When the true distribution is not in the model class, $B_{\phi^\star_\omega}$ converges to the closest representation of the true Bellman operator according to the parameterization that maximizes the likelihood $\mathbb{E}_{P_B(b,s,a|\omega)}[\log p(b,s,a|\phi)]$. This is analogous to frequentist convergent TD learning where the function approximator converges to a parameterization that minimizes the projection of the Bellman operator into the model class [Bhatnagar et al., 2009, Sutton et al., 2009a,b].

3.3.3 Gaussian BBO

Before we derive the approximate inference versions of BBO in sections 3.4.1 and 3.4.2, we introduce a nonlinear Gaussian model $P(b|s,a,\phi) = \mathcal{N}(\hat{B}_\phi(s,a), \sigma^2)$ that is commonly used for Bayesian regression [Gal, 2016, Murphy, 2012]. The mean of the model, $\hat{B}_\phi \approx \mathcal{B}[\hat{Q}_\omega]$, is a nonlinear function approximator that best represents the Bellman operator $\mathcal{B}[\hat{Q}_\omega]$ and the variance $\sigma^2 > 0$ represents the aleatoric uncertainty in our samples. Ignoring the log-normalization constant c_{norm}, the log-posterior is an empirical mean squared error between the empirical Bellman samples and the model mean $\hat{B}_\phi(s_i,a_i)$ with additional regularization R due to the prior (see appendix A.1.3 for a derivation):

$$-\log p(\phi|\mathcal{D}^N_\omega) = c_{\text{norm}} + \sum_{i=1}^{N} \frac{(b_i - \hat{B}_\phi(s_i,a_i))^2}{2\sigma^2} + R(\phi) \tag{3.7}$$

Finding a MAP estimate of the posterior corresponds to minimizing the empirical TD-target objective commonly used by frequentist approaches with target parameters, with an additional regularization term $R(\phi)$ that is controlled with

σ. Fellows et al. [2021] show that the asymptotic MSBBE minimizer under the Gaussian regime parameterizes a TD fixed point, should it exist. Furthermore, they also prove that the Bayesian approaches under the BBO framework and frequentist TD approaches converge to identical solutions.

3.4 Approximate BBO

Equation (3.6) demonstrates that minimizing MSBBE is feasible with a standard stochastic gradient descent algorithm, provided that sampling from the posterior distribution is tractable. Our focus in this thesis is on the development of control algorithms for complex continuous control systems. In such settings, employing expressive nonlinear function approximators is desirable, yet introduces complexities to the inference as the posterior normalization becomes intractable. In this section, we look into methods for learning a tractable posterior approximation $\hat{P}(\cdot|\mathcal{D}_\omega^N) \approx P(\cdot|\mathcal{D}_\omega^N)$. Concretely, these methods provide a tractable approximation to the MSBBE:

$$\text{MSBBE}_N(\omega) = \left\|\hat{Q}_\omega - \mathcal{B}_{\omega,N}\right\|_{\rho,\pi}^2 \approx \left\|\hat{Q}_\omega - \int \hat{B}_\phi \, d\hat{P}_\Phi(\phi|\mathcal{D}_\omega^N)\right\|_{\rho,\pi}^2 \tag{3.8}$$

3.4.1 Maximum a Posteriori BBO

Perhaps the most straightforward method for estimating the posterior within Bayesian frameworks is through the Maximum A Posteriori (MAP) estimation. This can be formally expressed as:

$$\begin{aligned} \hat{P}_\Phi(\cdot|\mathcal{D}_\omega^N) &= \delta_{\psi^\star(\mathcal{D}_\omega^N)}, \\ \text{where} \qquad \psi^\star\left(\mathcal{D}_\omega^N\right) &\in \arg\max_{\phi\in\Phi} \log p(\phi|\mathcal{D}_\omega^N). \end{aligned}$$

The MAP estimation approach yields a singular point estimate of the posterior, inherently limiting its ability to present uncertainty. This limitation restricts the algorithm's capability for uncertainty-driven exploration. Nonetheless, MAP estimation remains useful for policy evaluation where the agent cannot influence the data collection process and thus exploration is unnecessary, as it provides

a principled way of incorporating prior knowledge and regularization into the posterior estimation.

Substituting the MAP posterior estimate into the MSBBE in equation (3.8), yields a MAP-approximate MSSBE objective:

$$\text{MSBBE}_{\text{MAP}}(\omega) \approx \left\| \hat{Q}_\omega - \hat{B}_{\psi^\star(\mathcal{D}_\omega^N)} \right\|_{\rho,\pi}^2 \tag{3.9}$$

When a fixed point $\hat{Q}_\omega = \hat{B}_{\psi_l^\star(\mathcal{D}_\omega^N)}$ exists, minimizing $\text{MSBBE}_{\text{MAP}}(\omega)$ is equivalent to finding $\omega_l^\star$ such that $\psi_l^\star(\mathcal{D}_\omega^N) = \omega_l^\star$. We make a simplifying assumption that $\hat{Q}_\omega$ and $\hat{B}_\phi$ share the function space, implying that $\Phi = \Omega$ and $\hat{Q}_\omega = \hat{B}_\omega \; \forall \; \omega \in \Omega$. Under this assumption, finding the minimum of equation (3.9) is equivalent to minimizing the following, simpler auxiliary objective:

$$\mathcal{U}(\omega; \psi^\star) = \left\| \omega - \psi^\star(\mathcal{D}_\omega^N) \right\|_2^2,$$

which has the advantage that deterministic gradient updates can be obtained. $\mathcal{U}(\omega; \psi^\star)$ can still provide an alternative auxiliary objective when a fixed point does not exist as the minimum of equation (3.9) has the same solution as minimizing $\mathcal{U}(\omega)$ from section 3.4.1 for sufficiently smooth $\hat{B}_\phi$.

With Gaussian likelihood, substituting the log-posterior from equation (3.7), the MAP can be seen to minimize the following objective:

$$\psi^\star(\mathcal{D}_\omega^N) \in \arg\min_{\phi \in \Phi} \left(\sum_{i=1}^{N} \frac{\left(b_i - \hat{B}_\phi\left(\mathbf{s}_i, \mathbf{a}_i\right) \right)^2}{2\sigma^2} + R(\phi) \right).$$

In order to minimize the approximate MSBBE in equation (3.8), we must solve a system of equations that forms the following bi-level optimization problem [Bard, 1991]:

$$\begin{aligned} \omega &\in \arg\min_{\omega \in \Omega} \mathcal{U}(\omega; \psi^\star) \\ \psi^\star &\in \arg\min_{\phi \in \Phi} \mathcal{L}(\phi, \omega) = \arg\min_{\phi \in \Phi} -\log(\phi | \mathcal{D}_\omega^N) \end{aligned}$$

Solving a bi-level problem is known to be NP-hard [Bard, 1991]. To address this challenge, we adopt a two-timescale stochastic gradient descent approach,

introducing a variable $\psi \in \Phi$ to track $\psi^\star(\mathcal{D}_\omega^N)$, which leads to the following updates:

$$\psi_{k+1} = \psi_k + \alpha_k \Bigg(\left(r + \gamma \hat{Q}_{\omega_k}(\mathbf{s}', \mathbf{a}') - \hat{B}_{\psi_k}(\mathbf{s}, \mathbf{a})\right) \nabla_\phi \hat{B}_{\psi_k}(\mathbf{s}, \mathbf{a}) \Bigg) + \sigma^2 \nabla_\phi R(\psi_k), \quad \text{(fast)} \quad (3.10)$$

$$\omega_{k+1} = \omega_k + \beta_k \left(\psi_k - \omega_k\right), \quad \text{(slow)} \quad (3.11)$$

with time scales α_k and β_k such that $\lim_{k\to\infty} {}^{\beta_k}/_{\alpha_k} = 0$. In theory, these updates require projection operators that project the parameters back onto their compact spaces, but we omit them for clarity as they are rarely required in practice.

As the parameters ω_k are updated on a slower timescale, they lag the parameters ψ_k. These parameters share a similar role to the *lagged critic* parameters that have been shown to stabilize many model-free TD algorithms [Mnih et al., 2015]. We refer to algorithms that minimize the MSBBE in this way as *gradient* BBO, as they take the posterior's dependence of $\hat{Q}_\omega$ into account. In our experiments (section 3.6), we also test a version, which we refer to as *direct* BBO, where we ignore this dependency by setting $\hat{B}_\phi = \hat{Q}_\omega$ and thus effectively ignoring the slow update.

3.4.2 Randomized Priors BBO

One of the primary advantages of adopting a Bayesian framework in RL is its systematic handling of uncertainty. Bayesian methods enable the development of algorithms that explore efficiently by leveraging their understanding of uncertainty within the MDP. While the MAP estimates introduced in the previous section provide a solid approach for policy evaluation, their inability to capture the posterior's uncertainty renders them ineffective in control scenarios. This necessitates a shift towards more advanced posterior approximation techniques in order to develop methods for control.

In this section, we present an approximate BBO using Randomized Priors [Osband et al., 2018, 2019b]. Although BBO is compatible with any tractable approximate inference approach, existing works show that RP uncertainty estimates are conservative [Ciosek et al., 2020, Pearce et al., 2019] with strong empirical

performance in RL [Osband et al., 2018, 2019b] for the Gaussian model that we focus on in this chapter.

As discussed in section 2.1.5, Randomized Priors injects noise into the MAP point estimate through the prior and thereby provides a tractable method for obtaining an approximation of a generally-intractable posterior and effectively characterizing uncertainty. Specifically, RP introduces a noise variable $\epsilon \in \mathcal{E}$ with distribution $P_E(\epsilon)$ where the density $p_E(\epsilon)$ has the same form as the prior.

The practical RP uses ensembling [Osband et al., 2018], where L prior randomizations $\mathcal{E}_L \coloneqq \{\epsilon_l\}_{l=1}^L$ are first drawn from P_E. To use RP for BBO, we write the Q-function approximator as an ensemble of L parameters $\Omega_L \coloneqq \{\omega_l\}_{l=1}^L$ where $\hat{Q}_\omega = \frac{1}{L}\sum_{l=1}^L \hat{Q}_{\omega_l}$ and assume a prior with a density of the form $p(\phi) \propto \exp(R(\phi))$. We find the solution to the prior-randomized MAP objective for each ensemble member $l \in \{1 : L\}$:

$$\psi_l^\star(\omega_l) \in \arg\min_{\phi\in\Phi} \mathcal{L}(\phi; \mathcal{D}_{\omega_l}^N, \epsilon_l), \tag{3.12}$$

$$\text{with } \mathcal{L}(\phi; \mathcal{D}_{\omega_l}^N, \epsilon_l) \coloneqq \frac{1}{N}\left(R(\phi - \epsilon_l) - \sum_{i=1}^N \log p(b_i|s_i, a_i, \phi)\right). \tag{3.13}$$

The RP solution $\psi_l^\star(\omega_l)$ has dependence on ω_l that mirrors the BBO posterior's dependence on ω. To construct the RP approximate posterior $\hat{P}(\phi|\mathcal{D}_\omega^N)$, we average the set of perturbed MAP estimates over all ensembles: $\hat{P}(\phi|\mathcal{D}_\omega^N) \coloneqq \frac{1}{L}\sum_{l=1}^L \delta(\phi \in \psi_l^\star(\omega_l))$. To sample from the RP posterior $\phi \sim \hat{P}(\cdot|\mathcal{D}_\omega^N)$, we sample an ensemble uniformly $l \sim \text{Unif}(\{1 : L\})$ and set $\phi = \psi_l^\star(\omega_l)$.

The RP approximate posterior $\hat{P}(\phi|\mathcal{D}_\omega^N)$ depends on the ensemble of Q-function approximators $\hat{Q}_{\omega_l}$ and, just like with MAP BBO, we must learn an ensemble of optimal parameterizations $\omega_l^\star$. Following the same procedure as with section 3.4.1, we substitute $\hat{P}(\phi|\mathcal{D}_\omega^N)$ for the true posterior in equations (3.1) and (3.4) to derive an approximate ensembled randomized-prior MSBBE:

$$\text{MSBBE}_{\text{RP}}(\omega_l) \coloneqq \|\hat{Q}_{\omega_l} - \hat{B}_{\psi_l^\star(\omega_l)}\|_{\rho,\pi}^2, \tag{3.14}$$

from which we can, again, derive a simplified auxiliary objective:

$$\mathcal{U}(\omega_l; \psi_l^\star) \coloneqq \|\omega_l - \psi_l^\star(\mathcal{D}_\omega^N)\|_2^2. \tag{3.15}$$

Optimizing the system of equations

$$\omega_l \in \arg\min_{\omega\in\Omega} \mathcal{U}(\omega; \psi_l^\star)$$
$$\psi_l^\star \in \arg\min_{\phi\in\Phi} \mathcal{L}(\phi; \mathcal{D}_{\omega_l}^N, \epsilon_l)$$

is exactly equivalent to finding a prior-randomized MAP solution for each ensemble member, which we solve with the same two-timescale gradient approach that was introduced for MAP BBO in section 3.4.1. Specifically, we introduce an ensemble of parameters $\Psi_L \coloneqq \{\psi_l\}_{l=1}^L$ to track $\psi_l^\star(\omega_l)$ and adopt a two-timescale gradient update for each $l \in \{1 : L\}$ on the RP objectives:

$$\psi_{l,k+1} = \psi_{l,k} - \alpha_k \nabla_{\psi_{l,k}} \left(R(\psi_{l,k} - \epsilon_l) - \log p(b|s, a, \psi_{l,k})\right), \qquad \text{(fast)} \qquad (3.16)$$

$$\omega_{l,k+1} = \omega_{l,k} - \beta_k(\omega_{l,k} - \psi_{l,k}), \qquad \text{(slow)} \qquad (3.17)$$

where α_k and β_k are asymptotically faster and slower step sizes respectively. From a Bayesian perspective, we are concerned with characterizing the uncertainty after a *finite* number of samples $N < \infty$ and hence (b_i, s_i, a_i) should be drawn uniformly from the dataset $\mathcal{D}_{\omega_l}^N$ to form estimates of the summation in equation (3.13). When compared to existing RL algorithms, sampling from $\mathcal{D}_{\omega_l}^N$ is analogous to sampling from a replay buffer [Mnih et al., 2015].

As ω_l are updated on a slower timescale, they lag the parameters ψ_l. When deriving a Bayesian actor-critic algorithm in section 3.5, we demonstrate that these parameters share a similar role to a *lagged critic*. There is no Bayesian explanation for these parameters under existing approaches: when applying approximate inference to the posterior over the Q-function to $P_{Q^\pi}(\cdot|s, a, \mathcal{D}^N)$, the RP solution $\psi_l^\star$ has no dependence on ω_l. Hence, minimizing $\mathcal{U}(\omega_l; \psi_l^\star)$ and the approximate MSBBE has an exact solution by setting $\omega_l^\star = \psi_l^\star$. In this case, $\hat{Q}_{\omega_l^\star} = \hat{B}_{\psi_l^\star}$, meaning that existing approaches do not distinguish between the Q-function and Bellman operator approximators.

3.5 Bayesian Bellman Actor-Critic

Having established a practical method for approximate inference and characterizing uncertainty in the BBO framework, this section introduces a Bayesian analog of the actor-critic framework for solving continuous control problems.

Our algorithm relies on posterior sampling [Osband et al., 2018, 2019b] as a tractable approach for exploration. To carry out posterior sampling, we sample an MDP $\phi \in \Phi$ from our posterior and then follow an optimal policy $\pi^\star_\phi \in \arg\max_\pi J(\pi|\phi)$ for that ϕ.

For a given MDP $\phi \sim P_\phi(\cdot|\mathcal{D}^N_\omega)$, drawn from the posterior, and assuming $\hat{Q}_{\omega^\star} = \mathcal{B}_{\omega^\star,N}$, we can write the Q-function in terms of the Bellman operator as:

$$\begin{aligned} Q^\pi(s,a,\phi) &= \int r + \gamma \hat{Q}_{\omega^\star}(s',a')\, dP^\pi(r,s',a'|s,a,\phi) \\ &= \int b\, dP(b|s,a,\phi) \\ &= \hat{B}_\phi(s,a), \end{aligned}$$

We can find an optimal policy $\pi^\star_\phi \in \arg\max_\pi J(\pi|\phi)$ by optimizing:

$$\begin{aligned} J(\pi|\phi) &= \int\int Q^\pi(s,a,\phi)\, d\pi(a|s)\, d\rho(s) \\ &= \int\int \hat{B}_\phi(s,a)\, d\pi(a|s)\, d\rho(s) \\ &= \mathop{\mathbb{E}}_{\rho,\pi}\left[\hat{B}_\phi\right]. \end{aligned}$$

Under our RP approximation the optimization becomes:

$$\arg\max_\pi J(\pi|\phi^\star(\epsilon;\mathcal{D}^N_\omega)) = \arg\max_\pi \mathop{\mathbb{E}}_{\rho,\pi}\left[\hat{B}_{\phi^\star(\epsilon;\mathcal{D}^N_\omega)}\right]. \tag{3.18}$$

Solving equation (3.18) within our RP setup adds another optimization level and leads to the following nested optimization problem:

$$\pi^\star \in \arg\max_\pi J(\pi;\psi^\star), \qquad \text{(Actor)} \tag{3.19}$$

$$\omega^\star \in \arg\min_\omega \mathcal{U}(\omega;\psi^\star), \qquad \text{(Target Critic)} \tag{3.20}$$

$$\psi^\star \in \arg\min_\psi \mathcal{L}(\psi;\mathcal{D}^N_\omega). \qquad \text{(Critic)} \tag{3.21}$$

To highlight the similarity between this system and the commonly used actor-critic algorithms [Konda and Tsitsiklis, 2000], we label each objective with its corresponding component in the actor-critic system. The approximate inference parameters $\psi^\star$ are analogous to the critic parameters, the auxiliary objective parameters $\omega^\star$ can be interpreted as target critic parameters which lag the critic parameters, while π is the actor.

In practice, we learn a parametric policy π_θ with parameters θ, for which the actor gradient is defined as:

$$\nabla_\theta J(\theta; \psi^\star) = \mathop{\mathbb{E}}_{\rho,\pi}\left[\nabla_\theta \log \pi_\theta(a|s)\hat{B}_\phi(s,a)\right]. \tag{3.22}$$

Finally, to account for the ensembling introduced for RP, we introduce an ensemble of L policies π_{θ_l} parameterized by $\Theta_L \coloneqq \{\theta_l\}_{l=1}^L$, matching the L critic and target ensemble components. Each π_{θ_l} optimizes the actor loss (equation (3.19)) with respect to its corresponding target critic parameters ψ_l: $\theta_l^\star \in \arg\max_\theta J(\theta|\psi_l)$.

We use the Gaussian BBO model introduced in section 3.3.3 with a Gaussian prior such that $R(\phi) = \frac{1}{2}\left\|\phi - \phi_0\right\|_2^2$, which results in the following log-posterior:

$$-\log p(\phi|\mathcal{D}_\omega^N) = c_{\text{norm}} + \sum_{i=1}^N \frac{\left(b_i - \hat{B}_\phi(s,a)\right)^2}{2\sigma^2} + \frac{1}{\sigma_0^2}\left\|\phi - \phi_0\right\|_2^2$$

Plugging these into the RP objectives from section 3.4.2, we get the following gradient updates for our objectives:

$$\psi_{l,k+1} = \psi_{l,k} - \alpha_k\Big(\left(\hat{B}_{\psi_{l,k}}(s,a) - (r + \gamma\hat{Q}_{\omega_{l,k}}(s',a'))\right)\nabla\hat{B}_{\psi_{l,k}}(s,a) + \frac{\sigma^2}{\sigma_0^2}\left(\psi_{l,k} - \epsilon_l - \phi_0\right)\Big), \qquad \text{(critic)} \tag{3.23}$$

$$\omega_{l,k+1} = \omega_{l,k} - \beta_k\left(\omega_{l,k} - \psi_{l,k}\right), \qquad \text{(target critic)} \tag{3.24}$$

$$\theta_{l,k+1} = \theta_{l,k} - \zeta_k\nabla_\theta \log \pi_{\theta_{l,k}}(a|s)\hat{B}_{\phi_{l,k}}(s,a), \qquad \text{(actor)} \tag{3.25}$$

where (s, a, s', r) are sampled from the dataset D and actions $a' \sim \pi_{\theta_l}(\cdot|s')$. The actor is updated on a slower timescale than the target critic, such that $\beta_k > \zeta_k$. Similarly, the target critic parameters ω_l are updated on a slower timescale than the critic parameters, which mimics the target critic parameter update schedule in frequentist approaches [Haarnoja et al., 2018d, Mnih et al., 2015].

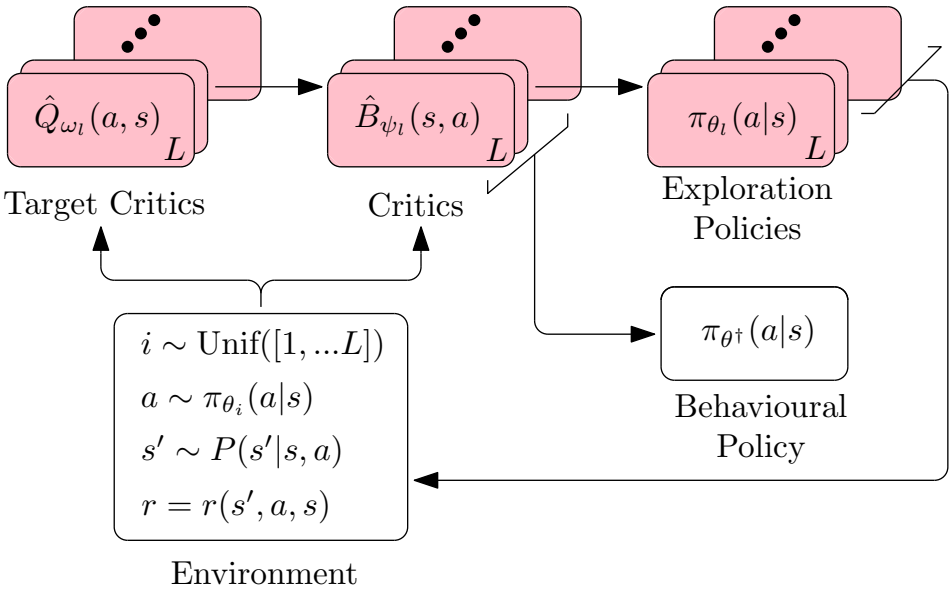


Figure 3.2: Schematic of Randomized-Prior Bayesian Bellman Actor-Critic (RP-BBAC).

The resulting algorithm, which we call Randomized-Prior Bayesian Bellman Actor-Critic (RP-BBAC), is listed in algorithms 1 to 3 and visually summarized in figure 3.2. The algorithm begins by initializing the model parameters, including the L prior randomizations $\mathcal{E}_L$.

RP-BBAC closely resembles the traditional actor-critic algorithms, as can be seen from the overall interaction loop between lines 4 and 14. One notable difference to existing frequentist approaches can be seen in line 5: at the start of each episode, RP-BBAC selects an exploratory actor in accordance with its current uncertainty in the MDP characterized by the approximate RP posterior. This encourages *deep exploration* (as discussed in section 2.2.5), where exploration not only considers immediate information gain but also the consequences of exploratory actions on future learning. RP-BBAC's exploration is also adaptive in that the actions from an exploration policy are directed towards minimizing epistemic uncertainty in the MDP and the posterior variance reduces as more data is collected.

Following N_{env} environment samples, the model posteriors are updated with algorithm 2. Mirroring a typical actor-critic setup, lines 7, 9 and 13 update the critic, critic target, and policy according to equations equations (3.23) to (3.25) from above.

Algorithm 1 RP-BBAC

Initialize $\Theta_L, \Omega_L, \Psi_L, \mathcal{E}_L, \theta^\dagger$ and $\mathcal{D} \leftarrow \varnothing$
Sample prior randomizations $\mathcal{E}_L \sim P_E$
Sample initial state $s \sim P_0$
while not converged **do**
  Sample policy $\theta_l \sim \text{Unif}(\Theta_L)$
  for $n \in \{1, ...N_{\text{env}}\}$ **do**
    Sample action $a \sim \pi_{\theta_l}(\cdot|s)$
    Observe next state $s' \sim P(\cdot|s, a)$
    Observe reward $r = r(s', a, s)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{s, a, r, s'\}$
  end for
  $\Theta_L, \Omega_L, \Psi_L \leftarrow$ UpdatePosterior$(\Theta_L, \Omega_L, \Psi_L, \mathcal{E}_L, \mathcal{D})$ ▷ Algorithm 2
  $\theta^\dagger \leftarrow$ UpdateBehavioralPolicy$(\theta^\dagger, \Psi_L, \mathcal{D})$ ▷ Algorithm 3
end while

Algorithm 2 UpdatePosterior$(\Theta_L, \Omega_L, \Psi_L, \mathcal{E}_L, \mathcal{D})$

$k \leftarrow 0$
while not converged **do**
  Sample mini-batch of transitions $T \sim \mathcal{D}$
  for $l \in \{1, ...L\}$ **do**
    for $\{s_i, a_i, r_i, s'_i\} \in T$ **do**
      $a'_i \sim \pi_{\theta_l}(\cdot|s'_i)$
      $\psi_l \leftarrow \psi_l - \frac{\alpha_k}{|T|} \nabla_{\psi_l} \hat{\mathcal{L}}^i_{\text{BBAC}}(\psi_l)$ ▷ Update critic (equation (3.23))
    end for
    $\omega_l \leftarrow \omega_l - \beta_k(\omega_l - \psi_l)$ ▷ Update critic target (equation (3.24))
    Sample batch B_l of states from $s \sim d(\cdot)$
    for $s \in B_l$ **do**
      $a \sim \pi_{\theta_l}(\cdot|s)$
      $\theta_l \leftarrow \theta_l + \frac{\zeta_k}{|B_l|} \nabla_{\theta_l} \pi_{\theta_l}(a|s) \hat{B}_{\phi_l}(s, a)$ ▷ Update policy (equation (3.25))
    end for
  end for
  $k \leftarrow k + 1$
end while

Finally, the exploration policies may not perform well at test time, so we learn a behavioral, or exploitation, policy $\pi_{\theta^\dagger}(a|s)$ parameterized by $\theta^\dagger \in \Theta$ from the data collected by our exploration policies using the ensemble of critics: $\{\hat{B}_{\psi_l}\}_{l=1:L}$. Theoretically, this is the optimal policy for the Bayesian estimate of the true MDP by using the approximate posterior to marginalize over the ensemble of Bellman operators. Furthermore, we augment our behavior policy objective with

Algorithm 3 UPDATEBEHAVIORALPOLICY($\theta^\dagger, \Psi_L, \mathcal{D}$)

$k \leftarrow 0$
while not converged **do**
  Sample mini-batch of states $B \sim \mathcal{D}$
  for $s \in B$: **do**
    $a \sim \pi(\cdot|s)$
    $\theta^\dagger \leftarrow \theta^\dagger + \frac{\alpha_k}{|B|} \nabla_{\theta^\dagger} \hat{J}(\theta^\dagger; s, a, \Psi_L)$ ▷ Equation (3.26)
  end for
  $k \leftarrow k + 1$
end while

entropy regularization:

$$\hat{J}(\pi|\Psi_L) = \mathop{\mathbb{E}}_{\rho,\pi}\left[\frac{1}{L}\sum_{l=1}^{L} \hat{B}_{\psi_l} + \alpha\mathcal{H}(\pi)\right], \tag{3.26}$$

allowing us to combine the exploratory benefits of posterior sampling with the faster convergence rates and algorithmic stability of regularized RL [Vieillard et al., 2020]. The full RP-BBAC algorithm is summarized in algorithm 1.

3.5.1 Related Work

Existing model-free Bayesian RL approaches assume either a parametric Gaussian [Fortunato et al., 2018, Gal and Ghahramani, 2016, Lipton et al., 2018, Osband et al., 2018, 2019b, Touati et al., 2019] or Gaussian process regression model [Engel et al., 2003, 2005]. Value-based approaches use the empirical Bellman function $b_\omega(s', a, s) = r(s', a, s) + \gamma \max_{a'} \hat{Q}_\omega(s', a')$ whereas actor-critic approaches use the empirical Bellman function $b_\omega(s', a', s, a) = r(s', a, s) + \gamma\hat{Q}_\omega(s', a')$. As we alluded to earlier, existing methods that use bootstrapping inadvertently approximate the posterior predictive over Q-functions with the BBO posterior predictive $P_{Q^\pi}(\cdot|s, a, \mathcal{D}^N) \approx P_B(\cdot|s, a, \mathcal{D}^N_\omega)$. These methods minimize an approximation of the MSBBE where the Bayesian Bellman operator is treated as a supervised target, ignoring its dependence on ω: gradient descent approaches drop gradient terms and fitted approaches iteratively regress the Q-function approximator onto the Bayesian Bellman operator $\hat{Q}_{\omega_{k+1}} \leftarrow \mathcal{B}_{\omega_k,N}$. In both cases, the updates may not be a contraction mapping for the same reasons as in non-Bayesian TD [Tsitsiklis

and Van Roy, 1997] and so it is not possible to prove general convergence. The additional Bayesian regularization introduced from the prior can lead to convergence, but only in specific and restrictive cases [Antos et al., 2007, 2008, Brandfonbrener and Bruna, 2019, Feng et al., 2019].

We emphasize the importance of sampling the exploration policy from the posterior (line 5). Many existing methods, even those employing ensembles, naïvely apply approximate inference to the Bellman error, treating $\mathcal{B}[Q^\pi(s,a)]$ and $Q^\pi(s,a)$ as independent variables [Fortunato et al., 2018, Gal and Ghahramani, 2016, Lipton et al., 2018, Touati et al., 2019]. This leads to unreliable uncertainty estimates as the Bellman error cannot correctly propagate the uncertainty [O'Donoghue et al., 2018, Osband et al., 2018, 2019b]. Osband et al. [2019b] demonstrate that this can cause uncertainty estimates of $Q^\pi(s,a)$ at some (s,a) to be zero and propose BootDQN+Prior as an alternative to achieve deep exploration. BBO does not suffer from this issue as the posterior characterizes the uncertainty in the Bellman operator directly.

One of our primary objectives is to develop a Bayesian algorithm for continuous control. Because of its reliance on the arg max operator, BootDQN+Prior is not applicable for continuous action spaces or large discrete action domains, as a nonlinear optimization problem must be solved every time an action is sampled. Similarly, Bayesian Q-Learning [Dearden et al., 1998] applies only to discrete environments with tabular value functions. In contrast, our RP-BBAC algorithm serves as a continuous-action analog to BootDQN+Prior, maintaining Bayesian principles for exploration while scaling to more complex action domains.

3.6 Experiments

3.6.1 Convergent Nonlinear Policy Evaluation

To provide evidence for the convergence and consistency results under posterior approximation, we evaluate BBO in several nonlinear policy evaluation experiments that are constructed to present a convergence challenge to TD algorithms. As the policy is fixed for these experiments, there is no benefit to characterizing uncertainty

in the MDP, hence all implementations use the MAP updates, introduced in section 3.4.1, with a truncated Gaussian prior $p_\Phi(\phi) \propto \frac{1}{2\sigma_0^2} \|\phi - \phi_0\|_2^2$, resulting in the following updates:

$$\psi_{k+1} = \psi_k + \alpha_k \Big(\Big(r + \gamma \hat{Q}_{\omega_k}(\mathbf{s}', \mathbf{a}') - \hat{B}_{\psi_k}(\mathbf{s}, \mathbf{a}) \Big) \nabla_\phi \hat{B}_{\psi_k}(\mathbf{s}, \mathbf{a}) \Big) + \frac{\sigma^2}{\sigma_0^2}(\psi_k - \phi_0), \quad \text{(fast)} \quad (3.27)$$

$$\omega_{k+1} = \omega_k + \beta_k (\psi_k - \omega_k). \quad \text{(slow)} \quad (3.28)$$

We set ϕ_0 to our initial estimate of the value function parameters, which is initialized using a Glorot uniform initialization. All practical details of these experiments are found in appendix A.2.2.

We begin by verifying the convergence of nonlinear Gaussian BBO in the famous counterexample task of Tsitsiklis and Van Roy [1997], in which the `TD(0)` algorithm is provably divergent. We also include results for the frequentist nonlinear `TDC` and `GTD2` [Bhatnagar et al., 2009], which extend the convergent linear `TDC` and `GTD2` algorithms [Sutton et al., 2009a] to nonlinear function approximators by using a linear model with a basis vector that is a local linearization of the value function. The results are presented in figure 3.3, where we plot the mean squared error between the true value function and the approximate value function vs. training time steps. As expected, `TD(0)` diverges, while BBO converges to the optimal solution in the same way as convergent frequentist nonlinear `TDC` and `GTD2` [Bhatnagar et al., 2009].

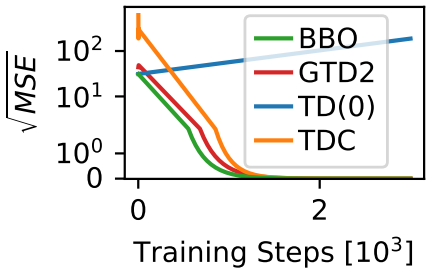


Figure 3.3: Learning curves in the Tsitsiklis counterexample.

Most interesting real-world tasks demand the use of expressive function approximators such as neural networks. Despite their lack of theoretical convergence guarantees, neural networks have been successfully used in practice for estimating value functions in a wide range of recent reinforcement learning applications. The gradient BBO algorithms can be shown to be a provably convergent method

for policy evaluation with runtime complexity that is linear in the dimension of the parameter space [Fellows et al., 2021]. In this experiment, we evaluate the convergence properties empirically by applying BBO to a nonlinear regime with neural network function approximators.

We consider three policy evaluation tasks commonly used to test convergence of nonlinear TD using neural network function approximators: `100-Link Pendulum` [Dann et al., 2014], `Puddle World` [Boyan and Moore, 1995], and `Mountain Car` [Boyan and Moore, 1995]. As we alluded to earlier in section 3.4.1, we refer to algorithms that update the parameters according to both the fast and slow update rules in equations (3.27) and (3.28) as *gradient* BBO as they take the posterior's dependence of $\hat{Q}_\omega$ into account and thus minimize the MSBBE exactly. We also test a version, which we refer to as *direct* BBO, where we ignore this dependence by setting $\hat{B}_\phi = \hat{Q}_\omega$ and thus effectively ignoring the slow update in equation (3.28). This corresponds to the update that existing model-free Bayesian RL approaches would take when they apply bootstrapping to their samples.

Our experiments are designed to (1) verify BBO's convergence and consistency properties in nonlinear regime, especially in cases where not all theoretical assumptions are fulfilled exactly, (2) to investigate the effect of sampling from the BBO posterior (gradient methods) vs sampling from the posterior over Q-functions with bootstrapping (direct methods), and (3) investigate the effect of additional regularization in BBO due to the prior.

Results

The results are presented in figure 3.4, which reports the root of the mean-squared error $\text{MSE}(\omega) \coloneqq \|V^\pi - \hat{V}_\omega\|_2^2$ between the true value function V^π and the learned value function $\hat{V}_\omega$. Gradient BBO with prior quickly converges to a good solution in all tasks, outperforming the non-Bayesian and direct BBO variants as well as other nonlinear methods (note that direct BBO without prior corresponds to vanilla `TD(0)`) across all tasks. Given that both the bi-level optimization and the prior

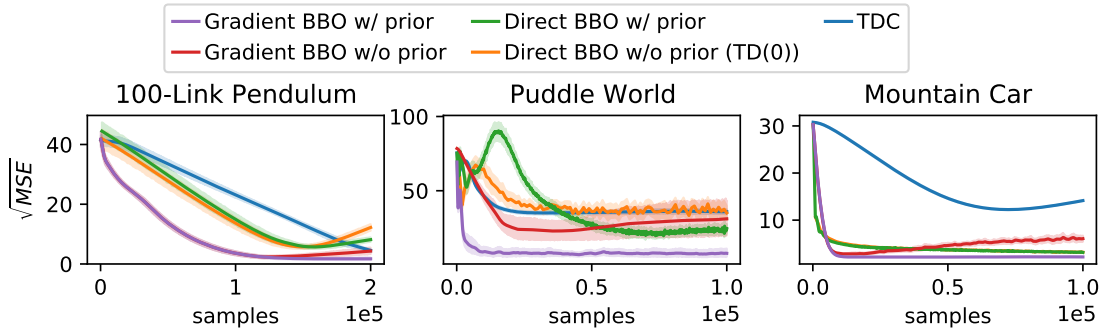


Figure 3.4: BBO: Nonlinear policy evaluation results.

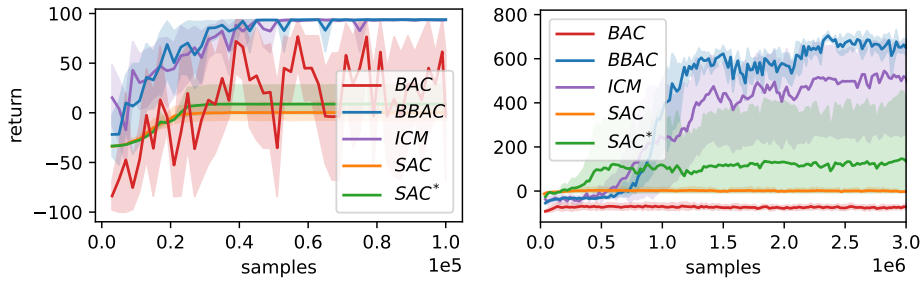
alone perform worse than Bayesian gradient BBO, the speed and quality of the solution can be attributed to their combination.

Furthermore, while not a direct measure, the convergence to near-zero MSE in `100-Link Pendulum` and `Mountain Car` tasks provides empirical evidence of the algorithm's consistency with the frequentist methods. An observant reader might wonder where the residual error in the `Puddle World` task comes from. We briefly discuss this in appendix A.2.3.

We thus conclude that i) by ignoring the posterior's dependence on ω, existing model-free Bayesian approaches are less stable and perform poorly in comparison to the gradient-based MSBBE minimization approach in equation (3.6), ii) regularization from a prior can improve performance of policy evaluation by aiding the optimization landscape [Du et al., 2017], and iii) better solutions in terms of mean squared error can be found using BBO instead of the local linearization approach of nonlinear `TDC/GTD2` [Bhatnagar et al., 2009].

3.6.2 Exploration for Continuous Control

We now evaluate RP-BBAC in continuous control tasks to showcase the benefits of BBO. We have two main goals for these experiments. First, we test the performance of algorithms derived from the BBO framework in continuous control tasks, where success requires solving a challenging deep exploration problem. Second, we investigate the empirical implications of ignoring the dependence of the TD error



(a) MountainCar **(b)** Cartpole

Figure 3.5: Learning curves in continuous control tasks with sparse rewards.

on the function approximator within the bootstrapping process of existing model-free Bayesian RL. Additionally, we investigate scenarios in which our RP-BBAC algorithm underperforms or fails to accomplish the tasks, highlighting the algorithm's limitations in more challenging settings. All practical details of our experiments can be found in appendix A.3.

Lower Dimensional Sparse-Reward Tasks.

In many benchmark tasks for continuous RL, such as the locomotion tasks from MuJoCo Gym suite [Brockman et al., 2016], the environment reward is shaped to provide a smooth gradient towards a successful task completion and naïve Boltzmann dithering exploration strategies from regularized RL can provide a strong inductive bias. In practical real-world scenarios, dense rewards are difficult to specify by hand, especially when the task is learned from raw observations like images. To demonstrate the exploration benefits of RP-BBAC, we first focus on a set of continuous control tasks with sparse rewards as continuous analogues of the discrete experiments used to test BootDQN+Prior [Osband et al., 2018]: `MountainCar-Continuous-v0` from Gym benchmark suite and a slightly modified version of the `cartpole-swingup_sparse` from DeepMind Control Suite [Tassa et al., 2018]. Both environments have a sparse reward signal and penalize the agent proportional to the magnitude of executed actions. As the agent is always initialized in the same state, it has to deeply explore costly states in a directed manner for hundreds of steps until it reaches the rewarding region of the state space.

We compare RP-BBAC with two variants of the state-of-the-art Soft Actor-Critic (SAC): one that is the exact algorithm presented in [Haarnoja et al., 2018d]; SAC*, which is a tailored version of the original, which uses a single Q-function to avoid *pessimistic underexploration* [Ciosek et al., 2019] due to the use of the double-minimum-Q trick (see appendix A.3 for details); and third, ICM, which refers to SAC augmented with *intrinsic curiosity* [Pathak et al., 2017] to provide a baseline of a modern exploration algorithm. Additionally, to understand the practical consequences of ignoring the dependence of the TD error on the function approximator within the bootstrapping process of existing Bayesian model-free methods, we also compare with BAC which is a variant of RP-BBAC where $\hat{Q}_{\omega_l^\star} = \hat{B}_{\psi_l^\star}$, thereby ignoring the target critic updates in equation (3.24). As we discussed in section 3.4.2, BAC is the Bayesian Actor-Critic that results from applying RP approximate inference to the posterior over Q-functions used by existing model-free Bayesian approaches with bootstrapping, so its choice of a benchmark allows us to evaluate the empirical benefits of using BBO vs the existing model-free Bayesian RL framework.

The results are shown in figure 3.5. Due to the lack of smooth signal towards the task completion, SAC consistently fails to solve the tasks and converges to always executing the 0-action due to the action cost term, while SAC* achieves the goal in one out of five seeds thanks to a fortuitous parameter initialization. The performance of BAC is unstable and the algorithm fails to converge to an optimal policy. ICM, driven by intrinsic curiosity is similarly able to solve both of the tasks. Its slightly lower performance in the Cartpole task stems from one of the five random seeds that did not converge within the experiment window. This is likely attributable to learning instabilities in the underlying SAC rather than inadequate exploration, given that the underperforming seed still achieves the sparse reward. RP-BBAC similarly solves both tasks across all five seeds, providing evidence of its exploration capabilities.

To further understand the reasons behind the poor performance of BAC, SAC, and SAC*, we provide a state support analysis for `MountainCar-Continuous-v0`.

The plots in figure 3.7 confirm the deep, adaptive nature of RP-BBAC's exploration, which leads the agents to systematically explore regions of the state-action space with high uncertainty. In the beginning, even when the actions are costly and no positive rewards are encountered, there still exists ensemble members whose value are optimistic under uncertainty. At $T = 9000$, the agent has not yet discovered any reward from the environment, but the disagreement between ensemble members drives the algorithm to deeply explore the state space to reflect the uncertainty that the agent has over undiscovered states that could yield high returns. At around $T = 14000$ (not shown in the plot), the agent achieves the goal for the first time and the value functions start to shape towards the optimal solution, demonstrating that the posterior concentrates and thus limiting further exploration and preventing the agent from revisiting sub-optimal states.

The same analysis for SAC and SAC* confirms the inefficiency of the exploration typical of RL as inference: the agent avoids costly actions that would ultimately lead to rewarding states, thus repeatedly exploring actions that lead to poor performance and rarely explores beyond its initial state. This is corroborated by the value functions, which are prematurely driven to sub-optimal solutions. In comparison to RP-BBAC, SAC's exploration is myopic and is not adaptive, with the agent failing to explore beyond the sub-optimal solution that its value functions have converged to, repeatedly taking actions that, even by its own incorrect belief in the MDP are sub-optimal.

Finally, the state support analysis for BAC in appendix A.3.1 confirms that by using the posterior over Q-functions with bootstrapping, it cannot accurately capture the epistemic uncertainty in the MDP. Initially, exploration is similar to RP-BBAC: at $T = 9000$, the agent has not yet discovered any reward from the environment and the disagreement between ensemble members drives the algorithm to explore the state space similarly as with BBAC (see figure 3.7). However, as time progresses, the ensembles never concentrate with increasing number of samples, which can be attributed to the convergence issues that stem from ignoring the posterior's dependence on ω. Thus the approximate posterior cannot adequately

characterize the epistemic uncertainty and the residual stochasticity continues to drive exploration even when the agent should have learned to ignore actions that do not lead to increased returns. In comparison to SAC, the initial uncertainty estimates enable the BAC to reach the goal but, because the Q-function learning is unstable without target networks and the agent spends more time re-exploring around the origin than BBAC.

The results in figure 3.5 demonstrate that the theoretical issues with existing approaches have negative empirical consequences, verifying that it is essential for Bayesian model-free RL algorithms with bootstrapping to sample from the BBO posterior as BAC fails to solve both tasks where sampling from the correct posterior in RP-BBAC succeeds. In appendix A.3.2, we further investigate RP-BBAC's sensitivity to randomized prior hyperparameters. The range of working hyperparameters in these lower-dimensional tasks is wide and easy to tune.

We conclude from these experiments that: 1) due to its sophisticated, deep exploration capabilities, algorithms derived from the BBO framework can solve continuous control tasks at which state-of-the-art regularized actor-critic algorithms fail catastrophically; and 2) the issues associated with applying bootstrapping to existing model-free Bayesian approaches prevent an agent accurately quantifying uncertainty which has negative empirical impact on performance. This verifies that it is essential for Bayesian model-free RL algorithms with bootstrapping to sample from the BBO posterior.

Limitations in Higher-Dimensional Humanoid Task.

While our experiments on lower-dimensional tasks highlight the strengths of RP-BBAC, we also identified notable limitations when scaling to more complex, high-dimensional environments. To investigate this, we evaluate RP-BBAC and SAC in two variants of the OpenAI Gym [Brockman et al., 2016] `Humanoid-v3` task. In the *dense*-reward setting, we used the original environment that provides a dense set of shaping terms. In the *sparse*-reward version, we remove all the positive shaping

terms and disable the premature episode termination: the agent receives a positive reward only if it has moved at least five meters from the origin.

Dense-Reward Setting. In previous sections, we noted how SAC's use of the double-minimum-Q trick heavily hinders its exploration capabilities. In dense-reward environments, however, this pessimistic underexploration [Ciosek et al., 2020] provides a strong inductive bias that helps to avoid exploring wasteful areas of the state space. As can be seen in figure 3.6, SAC learns a stable and fast-running policy, achieving high returns, in just around 10^7 steps.

By contrast, RP-BBAC progresses much more slowly, barely making any progress after having learned to stay up. Its deep exploration in the high-dimensional state space becomes wasteful when the environment's dense reward already provides sufficient gradient information to guide learning. Consequently, RP-BBAC still focuses on exploration, struggling to match the performance that SAC achieves by exploiting the dense reward structure.

In theory, given enough training time, the RP-BBAC policy would likely converge to a high-value solution similar to SAC but this will likely take an infeasibly long time in practice. We note that there also exists a trade-off here: in principle, one could adjust RP-BBAC's parameters to reduce exploration. For example, using an ensemble size of one would make RP-BBAC very close to the original SAC (that does not rely on double-Q trick) [Haarnoja et al., 2018b]. However, we maintain an ensemble size of 8 to remain consistent with other tasks and to highlight the exploration-focused design of our approach.

Sparse-Reward Setting. In the sparse variant, neither SAC nor RP-BBAC demonstrate progress within the experiment's time horizon. Given SAC's myopic exploration, it predictably fails to discover the distant sparse reward. Although RP-BBAC is designed for deeper exploration, it also struggles to complete the task. While its exploration strategy occasionally uncovers novel states, even millions of steps are insufficient to reliably reach the distant reward in this high-dimensional domain. This negative result is consistent with prior observations: to the best of our

knowledge, no method at the time of original publishing of BBO had demonstrated success on such sparse humanoid task. More recent work [Tasdighi et al., 2024] has shown some progress in solving the task.

These results highlight two key points. First, in dense-reward tasks, RP-BBAC's deep exploration can be counterproductive, as it can over-explore and fail to leverage the guidance provided by rich and frequent reward signals. Second, truly sparse, high-dimensional tasks pose significant challenges for both SAC and RP-BBAC. Although RP-BBAC offers stronger directed exploration, the complexity and size of the humanoid state space, combined with sparse rewards, renders learning prohibitively difficult within a typical training budget. Addressing these challenges likely requires more sophisticated Bayesian exploration, improved approximate inference, larger compute budgets, and improved priors.

Overall, these findings emphasize that, while RP-BBAC excels at deep exploration in lower-dimensional tasks, additional research is required to address its limitations — and those of deep exploration methods in general — in high-dimensional, sparse-reward settings.

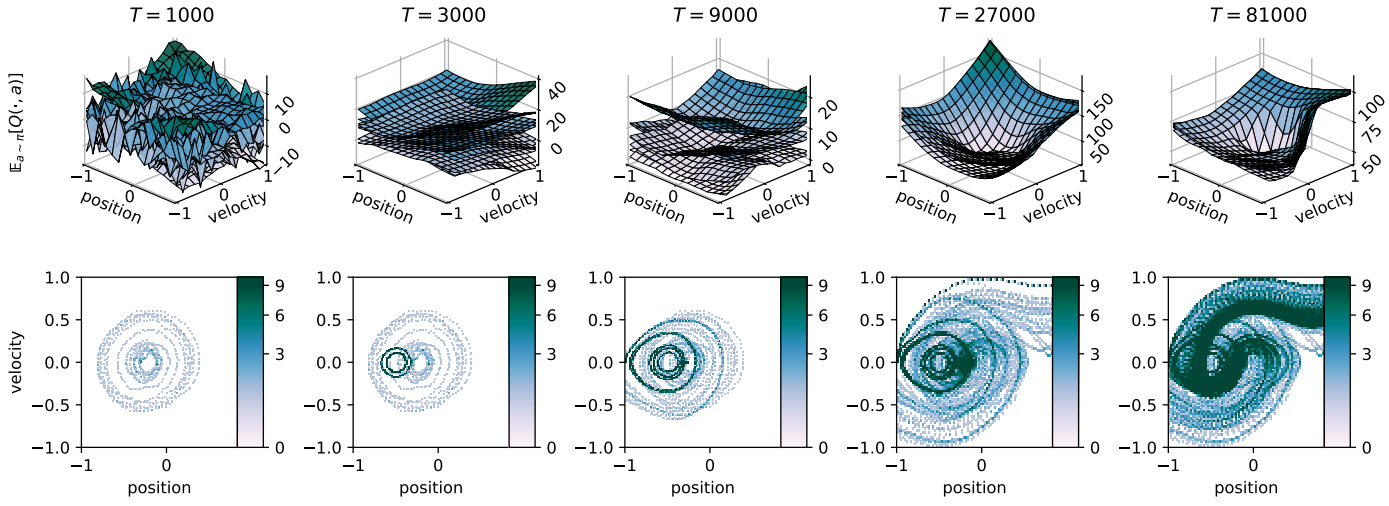


Figure 3.7: Diagnostics throughout learning of BBAC in `MountainCar-Continuous-v0` environment. (Top) Expected Q-values for each ensemble member ($L = 8$) over environment states $(position, velocity)$. (Bottom) State visitation plots show how the covered states evolve during learning. The deep, adaptive exploration carried out by BBAC leads to the agent systematically exploring regions of the state-action space eventually leading to successful task completion.

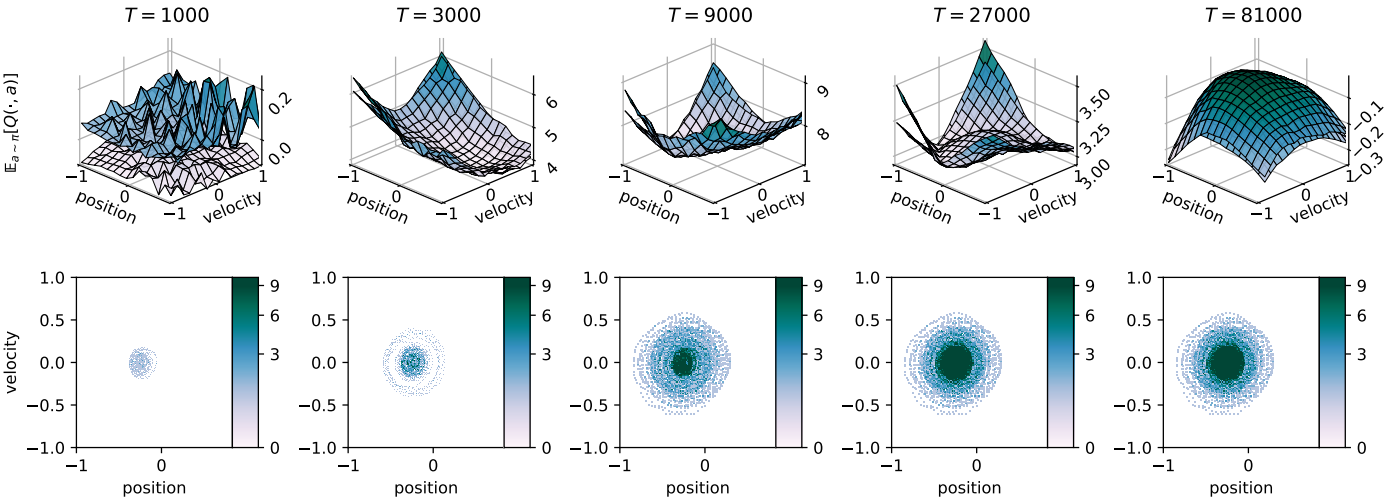


Figure 3.8: Diagnostics throughout learning of SAC in `MountainCar-Continuous-v0` environment. (Top) Ensemble values over environment state $(position, velocity)$. Notice the local maximum in the Q functions around the initial state at $T = 27000$. Similar, but more subtle maximum exists at $T = 9000$. (Bottom) State visitation plots show how the covered states evolve during learning. Due to the naïve exploration, SAC repeatedly explores actions that lead to poor performance and rarely explores beyond the initial state.

3.7 Conclusion

In this section, we introduced the *B*ayesian *B*ellman *O*perators (BBO) framework, to address the limitations of existing Bayesian model-free algorithms that use bootstrapping. Using this framework, we derived a practical policy evaluation method and introduced the Bayesian Bellman Actor-Critic (BBAC) algorithm for continuous control. Our experiments demonstrated the algorithm's consistency and convergence properties in the policy evaluation setting.

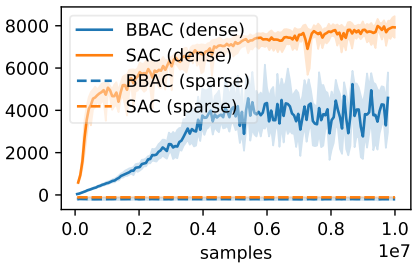


Figure 3.6: Learning curves in higher-dimensional `Humanoid-v3` task, with sparse (dashed lines) and dense (solid lines) rewards.

Similarly, in control setting, BBAC demonstrated deep and adaptive exploration capabilities driven by the agent's uncertainty of the environment. Our results are

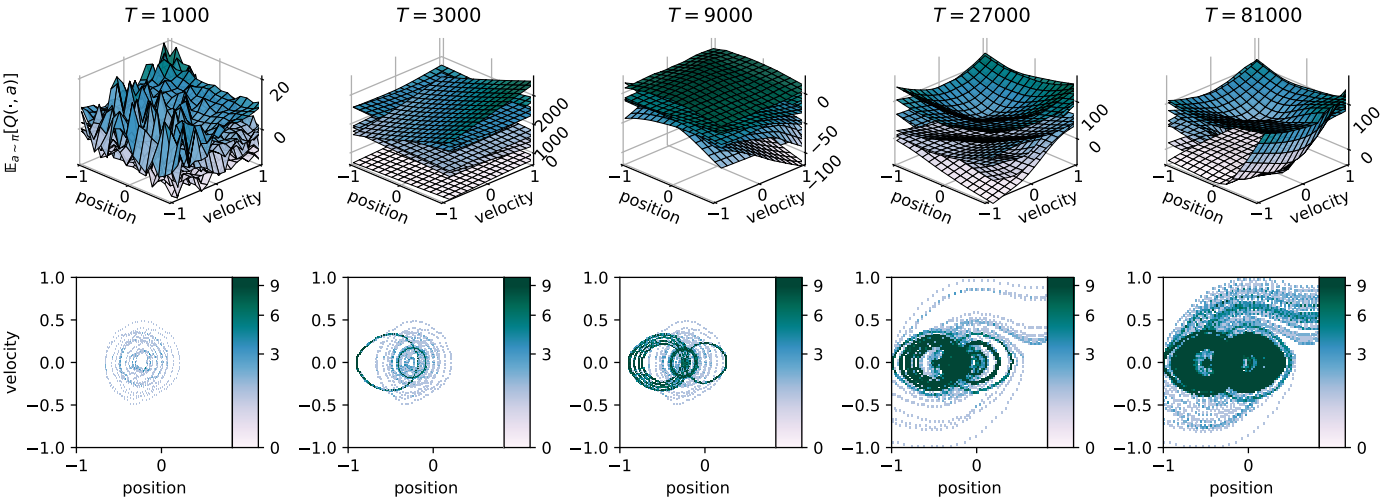


Figure 3.9: Diagnostics throughout learning of BAC in `MountainCar-Continuous-v0` environment. (Top) Expected Q-values for each ensemble member ($L = 8$) over environment states ($position, velocity$). (Bottom) State visitation plots show how the covered states evolve during learning. The agent is able to reach the goal but, due to unstable learning, spends more time exploring around the origin than BBAC.

encouraging and hopefully underscore the potential for wider adoption of model-free Bayesian paradigm for continuous RL applications.

Despite these promising outcomes, our experiments in higher-dimensional domains highlight that Bayesian exploration methods are not a panacea. Evidently, even truly Bayes-optimal strategies become intractably expensive to deploy in complex, tabula-rasa scenarios, especially without informative priors. To overcome these challenges, we will turn our focus in the next chapter to transfer learning techniques, which aim to incorporate additional sources of knowledge and thereby mitigate the inefficiencies of tabula-rasa learning.

4

Priors, Hierarchy, and Information Asymmetry for Skill Transfer in Reinforcement Learning

Contents

This chapter is based on [Salter, Hartikainen*, Goodwin, and Posner, 2022], co-authored with Sasha Salter, with equal contribution from both authors.*

4.1 Introduction

While Reinforcement Learning (RL) algorithms have recently achieved remarkable success across a range of domains [Lillicrap et al., 2015, Mnih et al., 2015, Silver et al., 2017], they remain sample inefficient [Abdolmaleki et al., 2018, Haarnoja et al., 2018b] and are therefore of limited use for many real-world control applications. To efficiently tackle novel situations, intelligent agents discover and reuse skills at multiple levels of behavioral and temporal abstraction. For example, in manipulation domains, beneficial abstractions could include low-level instantaneous motor primitives as well as longer-horizon higher-level object manipulation strategies. Equipping lifelong reinforcement learners [Parisi et al., 2019] with the capability to learn throughout their lifespan and utilize such skill abstractions could significantly enhance their applicability in the real world.

To this end, two paradigms have recently been introduced. KL-regularized RL [Galashov et al., 2019, Teh et al., 2017] presents an intuitive approach for data-driven behavior transfer in multi-task learning. It facilitates skill reuse by regularizing policy behavior towards a task-agnostic prior, which is learned by distilling common behaviors across multiple tasks into it, thus encouraging the reuse of common behaviors across tasks. Concurrently, hierarchical RL also enables skill discovery [Haarnoja et al., 2018a, Hausman et al., 2018, Merel et al., 2020, Wulfmeier et al., 2020a,b] by considering a two-level hierarchy in which the high-level policy is task-conditioned, while the low-level remains task-agnostic. The lower level of the hierarchy therefore discovers skills that are transferable across tasks. Both behavior priors and hierarchy offer their own skill abstraction. However, when combined, hierarchical KL-regularized RL can discover multiple abstractions. While some attempts have been made to combine these methods [Goyal et al., 2019, Liu et al., 2022, Tirumala et al., 2019, 2020], the effectiveness of such integrations in enhancing transfer learning has been inconsistent, with some approaches, such as that by Tirumala et al. [2019], failing to show considerable performance improvements.

The success of transferring skills using the aforementioned hierarchical and KL-regularized RL methods is highly dependent on the appropriate choice of information

asymmetry (IA). IA involves strategically masking information across different parts of the architecture to encourage independent and, ideally, generalized behaviors across the masked dimensions [Galashov et al., 2019]. For instance, self-driving vehicles can learn universal skills by basing the prior only on the local proprioceptive information, thus remaining agnostic to global coordinates. Similarly, in robotic manipulation, ignoring certain object characteristics like shape or weight allows the robot to develop generalizable grasping techniques that are independent of these factors. The choice of IA influences the behaviors learned and their transferability. Prior studies have based their IA choices on intuition and domain-independent considerations, often exploring only a limited range of asymmetries (see table 4.1), which can limit the benefits of transfer when sub-optimally selected. We demonstrate that this indeed is the case for many methods [Bagatella et al., 2022, Galashov et al., 2019, Pertsch et al., 2021, Tirumala et al., 2019, 2020, Wulfmeier et al., 2020a] across our tested domains. Hence, a more systematic approach that is domain-dependent and grounded in theory and empirical data, is necessary for selection of IA to fully leverage skill transfer.

In this chapter, we employ hierarchical KL-regularized RL to effectively transfer skills across sequential tasks. We begin by theoretically and empirically exploring the crucial *expressivity-transferability* trade-off — controlled by choice of information asymmetry — of skills across sequential tasks for hierarchical KL-regularized RL. Our findings, supported by theoretical insights and empirical evidence, reveal how conditioning skill modules on either too much (e.g. on entire history) or too little (e.g. just the current observation) information can hinder transfer by leading to the discovery of either overly general (e.g. motor primitives) or overly specialized skills (e.g. non-transferable, task-specific behaviors). By evaluating a broad spectrum of information asymmetries between the hierarchical policy and the prior, we highlight the limitations of prior methods that selected sub-optimal IAs for our domains, significantly limiting transfer performance. In response, we introduce *A*ttentive *P*riors for *E*xpressive and Transferable *S*kills (APES), a

novel approach that forgoes user intuition and automates the choice of IA in a data-driven, domain-dependent manner.

APES builds on our expressivity-transferability framework to learn the relevant choice of information asymmetry between policy and prior. Specifically, APES conditions the prior on the entire history, allowing for *expressive* skills to be discovered, and learns a low-entropic attention-mask over the input, paying attention only where necessary, so as to minimize covariate shift and improve *transferability* across domains. Our experiments across domains of varying levels of sparsity and extrapolation, including a robot block stacking task, demonstrate how APES outperforms existing methods, while automating IA choice and bypassing arduous IA sweeps. Further ablations emphasize the benefits of combining hierarchy and priors for discovering expressive multi-modal behaviors.

4.2 Skill Transfer in Reinforcement Learning

We consider multi-task reinforcement learning in Partially Observable Markov Decision Processes (POMDPs), defined by $M_k = (\mathcal{S}, \mathcal{X}, \mathcal{A}, r_k, p, p_k^0, \gamma)$, with tasks k sampled from $p(\mathcal{K})$. $\mathcal{S}$, $\mathcal{A}$, $\mathcal{X}$ denote observation, action, and history spaces, with observations denoted as $\mathbf{s} \in \mathcal{S}$, actions as $\mathbf{a} \in \mathcal{A}$, and history of observations up to timestep t as $\mathbf{x}_t = (\mathbf{s}_0, \mathbf{a}_0, \mathbf{s}_1, \mathbf{a}_1, \ldots, \mathbf{s}_t)$. $p(\mathbf{x}'|\mathbf{x}, \mathbf{a}) : \mathcal{X} \times \mathcal{X} \times \mathcal{A} \to \mathbb{R}_{\geq 0}$ is the dynamics model. The reward function $r_k : \mathcal{X} \times \mathcal{A} \times \mathcal{K} \to \mathbb{R}$ is assumed to be history-, action- and task-dependent.

4.2.1 KL-Regularized Reinforcement Learning

The typical multi-task KL-regularized RL objective [Kappen et al., 2012, Rawlik et al., 2012, Schulman et al., 2017, Todorov, 2007] takes the form:

$$\mathcal{J}(\pi, \pi_0) = \mathop{\mathbb{E}}_{\substack{\tau \sim p_\pi(\tau), \\ k \sim p(\mathcal{K})}} \left[\sum_{t=0}^{\infty} \gamma^t \Big(r_k(\mathbf{x}_t, \mathbf{a}_t) - \alpha_0 \mathrm{D}_{\mathrm{KL}}(\pi(\mathbf{a}|\mathbf{x}_t, k) \parallel \pi_0(\mathbf{a}|\mathbf{x}_t)) \Big) \right] \tag{4.1}$$

where γ is the discount factor and α_0 weighs the individual objective terms. π and π_0 denote the task-conditioned policy and task-agnostic prior respectively.

The expectation is taken over tasks k and trajectories τ induced by the policy π and initial observation distribution. Summation over t occurs across all episodic timesteps. When optimized with respect to π, this objective can be viewed as a trade-off between maximizing rewards while remaining close to trajectories produced by π_0. When π_0 is learned, it can learn shared behaviors across tasks and bias multi-task exploration [Teh et al., 2017]. We consider the sequential learning paradigm, where skills are learned from past tasks, $p_{\text{source}}(\mathcal{K})$ and leveraged while attempting the transfer set of tasks, $p_{\text{transfer}}(\mathcal{K})$.

The interpretation of KL divergence direction depends on which policy we optimize. When learning the prior policy π_0, this loss can be seen as

The Kullback–Leibler divergence can be viewed in two directions, often called *forward* KL and *reverse* KL, which differ in how they measure the discrepancy between two probability distributions. Assuming a fixed distribution p and a learned distribution q_θ, these directions are defined as follows:

$$\begin{aligned} &\text{Forward KL: } \mathrm{D}_{\mathrm{KL}}(p \parallel q_\theta) \\ &\text{Reverse KL: } \mathrm{D}_{\mathrm{KL}}(q_\theta \parallel p) \end{aligned} \tag{4.2}$$

These two directions capture different properties of the distribution. The forward KL term is "mode-covering": it penalizes the distribution q_θ for failing to cover modes that p assigns probability to. This tends to make q_θ spread its probability mass to cover all modes of p (avoiding zeros where p has support), often yielding a broader, mass-covering behavior. In practice, minimizing forward KL can lead to mean-seeking solutions that place non-zero probability even in intermediate regions between modes (to ensure all of p's support is covered). By contrast, the reverse KL regularizer is "mode-seeking": it penalizes q_θ for assigning mass to actions where p has little support, effectively pushing q_θ to concentrate on a single high-probability mode of p. This mode-seeking behavior prevents q_θ from drifting into low-probability regions of p, thus avoiding the mode-averaging effect.

In the context of KL-regularized reinforcement learning, when specifically learning the task-agnostic prior π_0 from task-specific policy distribution, the KL term in equation (4.1) operates in the forward direction, pushing π_0 to cover all

behavioral modes observed across different tasks. However, this coverage comes with limitations: if the task context is ambiguous given current observations, the prior may converge to an average behavior, at worst learning highly-suboptimal behaviors between the modes of optimal task-specific distributions. For example, if the optimal policy is dependent on task identifier (e.g., moving left vs. right), a prior that lacks this context conditioning might learn an intermediate behavior that is suboptimal for both tasks. Thus, when task-specific dynamics cannot be identified, the KL regularizer may keep the policy close to a suboptimal default strategy.

Another limitation arises when the chosen prior π_0 fails to capture the true shared structure among tasks, since KL regularization is only as helpful as the prior is informative. A good prior must be general enough to capture solutions for many tasks of interest while also being restrictive enough to provide meaningful guidance. If π_0 has poor coverage of the task-specific policies (e.g. missing important behaviors or modes), then penalizing divergence from π_0 can indeed harm learning; the agent might be discouraged from exploring outside the prior's support, even if the new task demands novel actions. Conversely, if π_0 is too broad or uninformative (assigning nearly equal probability to all behaviors, lacking any informative structure), the KL penalty becomes ineffective, offering little more than just a simple entropy regularizer. In extreme cases where tasks share almost no common structure, a fixed prior will either be too generic to aid learning or will impose an inappropriate bias.

4.2.2 Hierarchical KL-Regularized Reinforcement Learning

While KL-regularized RL has achieved success in various settings [Abdolmaleki et al., 2018, Haarnoja et al., 2018a, Pertsch et al., 2020, Teh et al., 2017], Tirumala et al. [2019] recently introduced a hierarchical extension to it. In their model, the policy π and prior π_0 are augmented with latent variables $\mathbf{z}$:

$$\pi(\mathbf{a}, \mathbf{z}|\mathbf{x}, k) = \pi^H(\mathbf{z}|\mathbf{x}, k)\pi^L(\mathbf{a}|\mathbf{z}, \mathbf{x})$$

$$\pi_0(\mathbf{a}, \mathbf{z}|\mathbf{x}) = \pi_0^H(\mathbf{z}|\mathbf{x})\pi_0^L(\mathbf{a}|\mathbf{z}, \mathbf{x}),$$

where H and L indicate the higher and lower hierarchical levels. This structure encourages the discovery of task-agnostic primitives by the shared low-level policy

($\pi^L = \pi_0^L$), while the high-level policy focuses on higher-level, task-specific skills. By not conditioning the high-level prior on the task ID, the model encourages reuse of common high-level abstractions across tasks. Tirumala et al. [2019] also propose the following upper bound for approximating the KL-divergence between the hierarchical policy and prior:

$$\begin{aligned} \mathrm{D}_{\mathrm{KL}}(\pi(\mathbf{a}|\mathbf{x}) \parallel \pi_0(\mathbf{a}|\mathbf{x})) \leq \mathrm{D}_{\mathrm{KL}}(\pi^H(\mathbf{z}|\mathbf{x}) \parallel \pi_0^H(\mathbf{z}|\mathbf{x})) \\ + \mathbb{E}_{\pi^H}\left[\mathrm{D}_{\mathrm{KL}}(\pi^L(\mathbf{a}|\mathbf{x},\mathbf{z}) \parallel \pi_0^L(\mathbf{a}|\mathbf{x},\mathbf{z}))\right]. \end{aligned} \tag{4.3}$$

This bound is agnostic to task conditioning and explicitly declared shared modules. While in principle this approach can be used to learn a high-level behavioral prior, π_0^H, in practice Tirumala et al. [2019] do not observe benefits from learning it.

The benefit of hierarchical reinforcement learning fundamentally relies on the assumption that a given task exhibits hierarchical structure, for example, temporally through options and skills, or structurally via task decomposition and multi-modal subtasks. Temporal abstraction assumes that decision-making problems can be effectively simplified by grouping primitive actions into temporally extended skills, thus improving exploration and potentially accelerating learning. Similarly, structural hierarchy presupposes that complex tasks can be naturally decomposed into distinct subtasks or modalities, thereby enabling modular learning and improved interpretability. Both forms of hierarchy have strong conceptual motivations inspired by natural cognition, where hierarchical control structures are evident in biological agents performing complex behaviors [Botvinick et al., 2009].

However, hierarchy is not always beneficial; its effectiveness is conditional on the compatibility between the assumptions and the environment's actual structure. When this assumption aligns with the task structure (such as in environments where subtasks are well-defined or repetitive) the hierarchical approach can significantly enhance efficiency, exploration, and transferability. Conversely, when the assumptions are not satisfied, hierarchical methods may impose unnecessary overhead, constrain the policy space, and ultimately yield suboptimal outcomes compared to flat, non-hierarchical methods. Thus, recognizing hierarchy as a

Table 4.1: Information Asymmetries Explored by Prior Works. $\mathbf{a}_{t-n:t}$ denotes the action history extending n steps back in the past.

Source	$\pi_0^{(H)}$ input
[Liu et al., 2022, Tirumala et al., 2020]	$\mathbf{x}_t$
[Bagatella et al., 2022]	$\mathbf{a}_{t-n:t}$
[Bagatella et al., 2022]	$\mathbf{x}_{t-1:t}$
[Ajay et al., 2020, Pertsch et al., 2020, 2021]	$\mathbf{s}_t$
[Rao et al., 2021, Tirumala et al., 2019, 2020]	$\mathbf{z}_{t-1}$
[Goyal et al., 2019, Tirumala et al., 2019]	–

powerful yet domain-dependent inductive bias is essential for effectively leveraging hierarchical reinforcement learning in practice.

4.2.3 Information Asymmetry

Information Asymmetry (IA) plays a key role in promoting the discovery of generalizable behaviors within both hierarchical KL-regularized RL and its conventional counterparts. IA can be understood as selectively masking information from certain modules. Not conditioning on specific environment aspects encourages independence and generalization across the modules [Galashov et al., 2019]. In the context of hierarchical KL-regularized RL, the explored asymmetries between the high-level policy π^H and prior π_0^H have been narrow [Ajay et al., 2020, Goyal et al., 2019, Liu et al., 2022, Pertsch et al., 2020, 2021, Rao et al., 2021, Tirumala et al., 2019, 2020]. Concurrent with our work, Bagatella et al. [2022] explores a wider range of asymmetries, closer to those we explore. We provide a summary of these asymmetries in table 4.1.

Choice of information conditioning heavily influences the discoverability and transferability of skills. For example, Pertsch et al. [2020] were able to identify observation-dependent behaviors suitable for navigating mazes, yet are unable to learn history-dependent skills, such as those that prevent revisiting the same paths multiple times. Conversely, the approach by Liu et al. [2022] condition on the state-action history and are able to learn these behaviors. However, our findings suggest that naïve conditioning on entire histories can be detrimental for skill

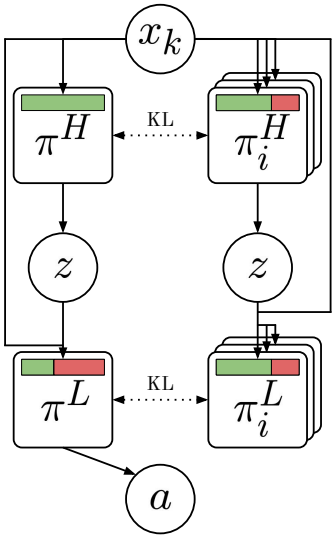


Figure 4.1: Hierarchical KL-regularized architecture. The hierarchical policy modules π^H and π^L are regularized against their corresponding prior modules π_i^H and π_i^L. The inputs to each module are filtered by an information gating function (IGF), depicted with colored rectangles, with green denoting passed-through and red masked information.

transfer, due to discovering behaviors that are overly specific and do not generalize favorably across different tasks. We refer to this dilemma as the *expressivity-transferability* trade-off. Crucially, prior approaches have pre-defined the asymmetry based on the practitioner's intuition, potentially compromising the optimal transfer of skills. By introducing theory behind the expressivity-transferability of skills, we present a simple data-driven method for automating the choice of IA, by learning it, which yields transfer benefits.

4.3 Model Architecture and Expressivity-Transferability Trade-Off

To investigate the roles of priors, hierarchy, and information asymmetry for skill transfer, it is crucial to analyze each mechanism in isolation, while also enabling the recovery of previous models of interest. To this end, we introduce a unified architecture, as shown in figure 4.1, featuring information gating functions (IGFs). These functions serve to decouple IA from the underlying architecture, thus allowing for a more nuanced control over what information each component receives. Each component has its own IGF, depicted with a colored rectangle in figure 4.1. Every

module is fed all environment information $\mathbf{x}_k = (\mathbf{x}, k)$ and distinctly chosen IGFs mask which part of the input each network has access to, thereby influencing which skills they learn. By presenting multiple priors, we enable a comparison with existing literature. With the right masking, one can recover previously investigated asymmetries [Bagatella et al., 2022, Goyal et al., 2019, Pertsch et al., 2020, Tirumala et al., 2019, 2020], explore additional ones, and also express purely hierarchical [Wulfmeier et al., 2020a] and KL-regularized models [Galashov et al., 2019, Haarnoja et al., 2018c].

4.3.1 Expressivity-Transferability Trade-Off

While existing works investigating the role of IAs for skill transfer in hierarchical KL-regularized RL have focused on multi-task learning [Galashov et al., 2019][1], we consider the sequential task setting, focusing on how well the prior π_0 adapts to covariate shifts. In contrast to multi-task learning, where the agent can access all tasks concurrently and thus faces only a gradual covariate shift, the sequential task setting introduces an additional challenge due to abrupt changes in both the task distribution, $p(\mathcal{K})$, and the trajectory distribution, $p_\pi(\tau)$. As such, it is important that the prior can handle such distributional and covariate shifts. Information asymmetry plays a crucial role in this, as we discuss next.

Theorem 4.3.1 (Covariate Shift)**.** *The more random variables a network depends on, the larger the covariate shift (here represented by KL-divergence) encountered across sequential tasks. That is, for two distributions p and q, and inputs $\mathbf{b}$ and $\mathbf{c}$ with $\mathbf{c} \subset \mathbf{b}$:*

$$\mathrm{D}_{\mathrm{KL}}(p(\mathbf{b}) \parallel q(\mathbf{b})) \geq \mathrm{D}_{\mathrm{KL}}(p(\mathbf{c}) \parallel q(\mathbf{c})).$$

Proof. See appendix B.1.1. □

In our context, p and q can be interpreted as representing the distribution of network inputs observed during training ($p_{\text{source}}(\cdot)$) and during transfer ($p_{\text{transfer}}(\cdot)$).

[1] Concurrent with our research, Bagatella et al. [2022] also investigated various IAs for sequential transfer.

Intuitively, this means that the more variables the network is conditioned on, the less likely it will transfer due to increased covariate shift encountered between source and transfer domains. This principle advocates for conditioning on a minimal set of necessary variables to facilitate transfer.

Consider, for instance, conditioning the high-level prior on the entire state-action history, $\mathbf{x}_{0:t}$, versus a more focused subset, $\mathbf{x}_{t-n:t}$, $n \in [0, t-1]$ (where $n \in [0, t-1]$). According to the principle outlined above with theorem 4.3.1, conditioning on a smaller subset of history results in a reduced covariate shift across sequential tasks, and this reduction is beneficial for the transferability of learned skills. That is,

$$\mathrm{D}_{\mathrm{KL}}(p_{\text{source}}(\mathbf{x}_{0:t}) \parallel p_{\text{transfer}}(\mathbf{x}_{0:t})) \geq \mathrm{D}_{\mathrm{KL}}(p_{\text{source}}(\mathbf{x}_{t-n:t}) \parallel p_{\text{transfer}}(\mathbf{x}_{t-n:t})).$$

Moreover, it is interesting to note that the covariate shift is upper-bounded by the overall shift in trajectories between the source and transfer tasks:

$$\mathrm{D}_{\mathrm{KL}}(p_{\pi_{\text{source}}}(\tau) \parallel p_{\pi_{\text{transfer}}}(\tau)) \geq \mathrm{D}_{\mathrm{KL}}(p_{\pi_{\text{source}}}(\tau_f) \parallel p_{\pi_{\text{transfer}}}(\tau_f)).$$

Here, p_π denotes the trajectory induced by policy π and τ_f, representing filtered trajectories such as $\tau_f = \mathbf{x}_{t-n:t} \subset \tau$ denotes the network's inputs after the application of its information gating function, and π_{source} and π_{transfer} the source domain and target domain policies. This suggests that reducing both trajectory and covariate shifts is crucial for enhancing the transferability of skills between sequential domains.

Nevertheless, the less information a prior is conditioned on, the less knowledge can be distilled and transferred:

Theorem 4.3.2 (Knowledge Distillation)**.** *The more random variables a network depends on, the greater its ability to distill knowledge in the expectation (reflected by the expected KL-divergence between the network's output distribution and a target distribution). Specifically, for a target distribution p and a network q with outputs* $\mathbf{a}$ *and possible inputs* $\mathbf{b}$*,* $\mathbf{c}$*,* $\mathbf{d}$*, such that* $\mathbf{b} = (b_0, b_1, ..., b_n)$*,* $\mathbf{d} \subset \mathbf{c} \subset \mathbf{b}$*,* $\mathbf{e} \in \mathbf{d} \setminus \mathbf{c}$*:*

$$\mathbb{E}_{q(\mathbf{e}|\mathbf{d})} \left[\mathrm{D}_{\mathrm{KL}}(p(\mathbf{a}|\mathbf{b}) \parallel q(\mathbf{a}|\mathbf{c}))\right] \leq \mathrm{D}_{\mathrm{KL}}(p(\mathbf{a}|\mathbf{b}) \parallel q(\mathbf{a}|\mathbf{d})).$$

Proof. See appendix B.1.2. □

In our context, p and q can be interpreted as policy distribution π and prior distribution π_0, with $\mathbf{a}$ representing an action $\mathbf{a}_t$, $\mathbf{b}$ the full state-action history $\mathbf{x}_{0:t}$, and variables $\mathbf{c}$, $\mathbf{d}$, and $\mathbf{e}$ subsets of the state-action history such that $\mathbf{e}$ consists of elements of $\mathbf{c}$ that are not in $\mathbf{d}$. Intuitively, theorem 4.3.2 indicates that, in the expectation, a more informed network can better approximate the target distribution's behavior and conditioning the prior on wider set of information facilitates more effective knowledge distillation between tasks.

Theorems 4.3.1 and 4.3.2 together suggest that information asymmetry results in an *expressivity-transferability* trade-off in skill acquisition. This trade-off highlights the balance between a network's ability to distill more knowledge (expressivity) and its capacity to effectively transfer this knowledge across tasks (transferability).

While information asymmetry influences *which* abstractions are discovered by hierarchy and priors, an important distinction must be made between *how* each approach transfers them. Hierarchy imposes hard constraints transferable behavior by enforcing the reuse of low-level action-abstractions π^L, whereas priors offer a softer transfer through KL-regularization, permitting policy deviations where beneficial. Consequently, managing covariate shift — and by extension, generalization — is more crucial for IA between hierarchical levels than between the policy and prior. This promotes reduced information conditioning for π^L than for π_0. Thus, in-line with previous works, in this work, we share the low-level policy π^L between policy and prior, and condition it only on state $\mathbf{s}_t$ thus ensuring minimal covariate shift and the discovery of instantaneous behaviors that generalize favorably. Unlike prior works, we consider conditioning the high-level prior, π_0^H, on additional information enabling richer behavior discovery and transfer. Specifically, we explore temporal conditioning on varying levels of histories $\mathbf{x}_{t-i:t}$ (with i denoting history depth), thus enabling priors to capture reusable, sequential, high-level behaviors across tasks.

We additionally explore the importance of hierarchy for increased prior expressivity. Here, hierarchy enables a richer prior distribution, capturing multi-modal behaviors arguably present in many real-world scenarios. Importantly, while increasing expressivity, hierarchy does not influence covariate shift or harm transferability.

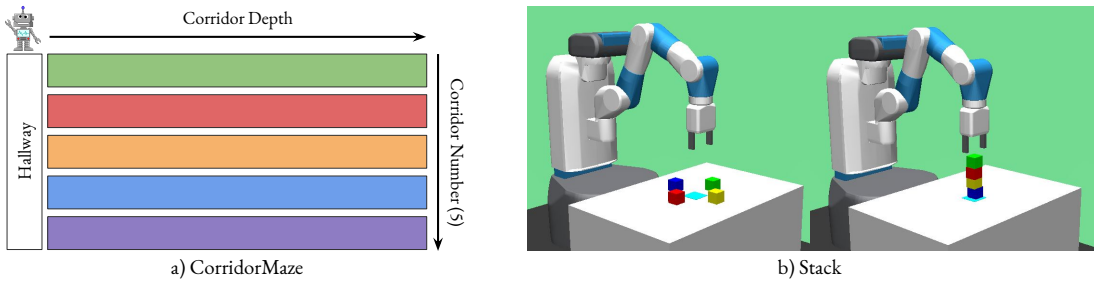


Figure 4.2: *Environments. a)* `CorridorMaze`*:* The agent starts in the hallway and must traverse a given sequence of corridors. The agent completes a corridor by traversing to its depth and back. *b)* `CubeStack`*:* The agent must stack the cubes in a given ordering over the light blue target pad.

We present further details on the architectural implications in appendix B.3. Next, we describe a data-driven method for learning information asymmetries.

4.4 APES: *A*ttentive *P*riors for *E*xpressive and Transferable *S*kills

Contrary to prior approaches that select information asymmetries based on intuition [Ajay et al., 2020, Bagatella et al., 2022, Galashov et al., 2019, Liu et al., 2022, Pertsch et al., 2020, Singh et al., 2020, Tirumala et al., 2019, 2020, Wulfmeier et al., 2020a], we introduce a novel, data-driven strategy that leverages learned Information Gating Functions (IGFs), introduced in section 4.3 and depicted in figure 4.1.

Existing methods achieve distinct information asymmetry through hard attention using predefined, static binary masks $\mathbf{m} \in \{0,1\}^{\dim(\mathbf{x}_k)}$ for filtering $\mathbf{x}_k$, where $IGF(\mathbf{x}_k) = \mathbf{m} \odot \mathbf{x}_k$, with $\odot$ denoting element-wise multiplication. We propose adopting learned attention masks and learning the masks based on an objective that integrates the hierarchical KL-regularized RL framework with a penalty on the attention masks to promote sparsity:

$$\mathcal{J}_{\text{APES}}(\pi, \pi_0, \{\mathbf{m}_i\}^{i\in I}) = \mathop{\mathbb{E}}_{\substack{\tau\sim p_\pi(\tau),\\ k\sim p(\mathcal{K})}} \left[\sum_{t=0}^{\infty} \gamma^t \Big(r_k(\mathbf{x}_t, \mathbf{a}_t) - \alpha_0 \mathrm{D}_{\mathrm{KL}}(\pi(\mathbf{a}|\mathbf{x}_t, k) \parallel \pi_0(\mathbf{a}|\mathbf{x}_t)) \Big)\right] - \sum_{i\in I} \alpha_{m_i} \mathcal{L}_{\text{ATTENTION}}(\mathbf{m}_i) \tag{4.4}$$

where α_{m_i} balances the standard KL-regularized RL objective with a sparsity penalty of each module's attention mask $\{\mathbf{m}_i\}^{i \in I}$. The attention masks span across all history dimensions, allowing the model to attend to both temporal and intra-observation and intra-action dynamics when needed, while the sparsity term, $\mathcal{L}_{\text{ATTENTION}}$, is designed to drive the attention masks towards sparse configurations.

Perhaps the simplest approach would be to either exhaustively enumerate or randomly sample these masks and carry out learning for each instance. While conceptually simple, this method would be difficult to apply in practice; as there is an exponential number ($2^{\dim(\mathbf{x}_k)}$ exactly) of possible masks, such a naïve approach would be computationally intractable for most interesting tasks, where the full action-state history can span tens or hundreds of timesteps, with each timestep itself being high-dimensional.

Instead of random sampling, we choose to learn the masks with gradient-based optimization. In order to learn a *hard* attention mask $\mathbf{m}_\mathbf{w}$ while maintaining differentiability, we use continuous relaxation of Bernoulli distribution [Jang et al., 2016, Maddison et al., 2016], $\text{RelaxedBernoulli}(\tau, \mathbf{w})$[2], parameterized with logits $\mathbf{w}$ and a fixed temperature parameter τ. The soft mask samples $\mathbf{m}_\mathbf{w}^{\text{SOFT}} \sim \text{RelaxedBernoulli}(\tau, \mathbf{w})$ provide a smooth approximation to the discrete mask, allowing us to backpropagate through them with respect to the logit parameter $\mathbf{w}$.

To enforce a hard masking of the policy inputs, we convert the soft sample $\mathbf{m}^{\text{SOFT}}$ into a thresholded, hard version of the mask, where each element at index i takes the value:

$$\mathbf{m}_i^{\text{HARD}} = \begin{cases} 1, & \text{if } 0.5 < \mathbf{m}_i^{\text{SOFT}} \\ 0, & \text{otherwise} \end{cases}$$

Since this thresholding is non-differentiable, we use the straight-through gradient estimator [Bengio et al., 2013]: during the backward pass, gradients are redirected through the soft values, while the forward pass uses the binary mask

$$\mathbf{m}^{\text{STRAIGHT-THROUGH}} = \mathbf{m}^{\text{SOFT}} + \text{sg}(\mathbf{m}^{\text{HARD}} - \mathbf{m}^{\text{SOFT}})$$

[2]Also referred to as the BinConcrete in [Maddison et al., 2016]

where sg denotes the stop-gradient-operation preventing the gradients flowing through its inputs. The elements of $\mathbf{m}^{\text{STRAIGHT-THROUGH}}$ are guaranteed to take values in $\{0, 1\}$ — thus either fully masking the inputs of the policy π^H or passing them directly through — while still allowing the gradients to propagate through the soft masks.

Finally, we define the attention penalty as the L_1-norm over the mask, encouraging the mask to zero out unnecessary inputs:

$$\mathcal{L}_{\text{ATTENTION}}(\mathbf{m}) \coloneqq \|\mathbf{m}\|_1 \tag{4.5}$$

This method, which we dub as *A*ttentive *P*riors for *E*xpressive and Transferable *S*kills (APES), encourages the discovery of *expressive* skills by allowing the model to focus on critical elements in the state-action history that are needed to optimize the hierarchical KL-regularized RL objective. Simultaneously, it encourages sparse, low-entropic attention masks $\mathbf{m}_i$, concentrating on critical environmental aspects necessary [Salter et al., 2022] for optimizing the main objective, thus minimizing covariate shift and improving the *transferability* of skills. By dynamically learning which aspects of the input to emphasize or ignore, APES balances the expressivity-transferability trade-off and thus aligns with the principles outlined in theorems 4.3.1 and 4.3.2, encouraging the discovery of skills that are both expressive and transferable.

In practice, we optimize the objective in equation (4.4) using an off-policy reinforcement learning algorithm similar to Soft Actor-Critic (SAC) [Haarnoja et al., 2018b], sampling experiences from the replay buffer, approximating agent returns with Retrace operator [Munos et al., 2016], and using double Q-learning for critic training [Hasselt, 2010]. More detailed explanation of the training process, architectural details, and hyperparameters used are provided in appendices B.2 and B.3.

4.4.1 Training Regime and The Information Asymmetry Setup

This subsection outlines the training regime for our experiments presented in section 4.5, with a focus on the role of priors, hierarchy, and information asymmetry

in facilitating skill transfer in sequential task learning. Specifically, we aim to apply skills learned from past tasks, $p_{\text{source}}(\mathcal{K})$, to a new set of transfer tasks, $p_{\text{transfer}}(\mathcal{K})$.

While one could investigate IA between hierarchical levels (π^H, π^L) as well as between policy and prior (π, π_0), we focus our exploration on the latter. Specifically, to ensure our study aligns with and extends those of existing literature [Tirumala et al., 2019, 2020, Wulfmeier et al., 2020a], we condition the low-level policy π^L on $\mathbf{s}_t$ and $\mathbf{z}_t$, and share it with the prior (i.e. $\pi^L = \pi_0^L$), to enable expressive multi-modal behaviors to be discovered with respect to $\mathbf{s}_t$. We are particularly interested in examining the role of IA between the high-level policy π^H and the high-level prior π_0^H in supporting expressive and transferable high-level skills between tasks. As is common, we assume that the source tasks are solved before addressing transfer tasks. Therefore, the source of skill acquisition — whether from demonstrations by a hard-coded expert or an optimal RL agent — is less critical to our core analysis. For simplicity, we discover skills and skill priors using variational behavioral cloning from expert policy π_e samples, formalized as follows:

$$\begin{aligned}&\mathcal{J}_{\text{BC}}(\pi, \pi_0, \pi_e, \{\mathbf{m}_{\mathbf{w}_i}\}^{i\in I}) = \\ &\sum_{j\in\{0,e\}} \mathbb{E}_{\substack{\tau\sim p_{\pi_e}(\tau),\\ k\sim p(\mathcal{K})}}\left[\sum_{t=0}^{\infty} -\alpha_j \mathrm{D}_{\text{KL}}(\pi(\mathbf{a}|\mathbf{x}_t, k) \parallel \pi_j(\mathbf{a}|\mathbf{x}_t))\right] - \sum_{i\in I} \alpha_{m_i}\mathcal{L}(\mathbf{m}_i)\end{aligned} \tag{4.6}$$

This formulation bears close similarity to the hierarchical KL-regularized RL objective in equation (4.4), with modifications to accommodate the specifics of imitation learning. Equation (4.6) effectively corresponds to equation (4.4) when we set $\gamma = 1$, $r_k(\cdot) = 0$, and use two specific priors, one learned (π_0) and one expert (π_e), and sample trajectories τ from the expert π_e rather than from the learned one π. For a more in-depth discussion on the similarities between the two objectives, see appendix B.2.2.

To evaluate distinct IAs' impact on skill transfer in a controlled way, we adopt the following two-stage training regime. In *stage (1)*, we train a single hierarchical policy π in a multi-task setup as per equation (4.6), but prevent gradient flow from prior π_0 to policy π. In *stage (2)*, we then transfer the skills by freezing the shared

modules (π^L, π_0^H) and train a new high-level policy π^H on the transfer domains using the hierarchical KL-regularized RL objective in equation (4.4).

We rely on the modularity assumption [Khetarpal et al., 2020a, Salter et al., 2022], which posits that low-level skills from source domains are sufficient for transfer tasks. While appearing restrictive, the more diverse the source domains are (commonly desired in settings like lifelong learning [Khetarpal et al., 2020a] and offline RL), the more probable it is that the optimal transfer policy can be obtained by recomposing the learned skills. Should this assumption not hold, one could alternatively, for example, fine-tune the low-level policy π^L during transfer — which would require tackling the catastrophic forgetting of skills [Kirkpatrick et al., 2017] — or train additional skills — e.g. by expanding $\mathbf{z}$ dimensionality for discrete $\mathbf{z}$ spaces. We leave this for future exploration. Extending APES to learning from sub-optimal demonstrations, possibly via advantage-weighted regression [Peng et al., 2019b] or other offline RL method [Levine et al., 2020] instead of behavioral cloning, also presents an interesting avenue for future research. Appendices B.2 and B.3 provide further details on our approach.

4.5 Experiments

Our experiments are designed to explore key aspects of sequential task learning, focusing on the interplay between hierarchy, priors, information asymmetry (IA), and their influence on skill transfer. We aim to answer the following questions: (1) Can we benefit from both hierarchy and priors for effective transfer? (2) How important is IA choice between high-level policy and prior and does it lead to *expressivity-transferability* trade-off? In practice, how detrimental is covariate shift for transfer? (3) How favorably does APES automate the choice of IA for effective transfer? Which IAs are discovered? To answer these questions, we investigate competitive skill transfer baselines, primarily those by Pertsch et al. [2020], Tirumala et al. [2019, 2020], Wulfmeier et al. [2020a], on similar navigation and manipulation tasks for which they were originally designed and tested against.

4.5.1 Environments

We evaluate on two domains (see figure 4.2): one designed for controlled investigation of core agent capabilities and another, more practical robotics domain. Both domains are characterized by modular behaviors whose discovery could yield transfer benefits. We briefly describe these environments below and, for the sake of clarity, leave the full details to appendix B.3.1.

`CorridorMaze`. In the `CorridorMaze` environment, the agent's goal is to traverse through a series of corridors, requiring completion of a full corridor cycle, i.e. reaching the end and returning to the origin, before proceeding to the next corridor. The environment is parameterized with N_c corridors, each of length N_l, and two different reward sparsity levels: *semi-sparse* rewarding for each half-corridor completion and *sparse* for rewarding the full task completion.

We parameterize the source tasks, $p_{\text{source}}(\mathcal{K})$, with $N_c = 2$, and consider two transfer tasks, $p_{\text{transfer}}(\mathcal{K})$: first one parameterized with $N_c = 2$ and sparse reward and the other one with $N_c = 4$ and semi-sparse reward. All the tasks use $N_l = 6$. In the transfer task with $N_c = 2$, the agent is required to traverse corridors that were already seen in the source task, except now in a different order. We call this the interpolated setting. The task with $N_c = 4$, on the other hand, includes corridors previously unseen by the prior, and we thus call it the extrapolated setting. These two settings allow us to investigate the generalization ability of different priors under varying levels of covariate shift.

For the skill acquisition phase, each prior is trained using 4e3 trajectories from a scripted policy traversing the source tasks ($p_{\text{source}}(\mathcal{K})$) in optimal order.

Appendix B.3.1 provides full definition on the environment transition functions and state, action, and observation spaces, as well as the expert policy used.

`CubeStack`. Our `CubeStack` domain modifies the 7-degree of freedom `FetchPickAndPlace-v0` environment [Plappert et al., 2018] available in the OpenAI Gym suite [Brockman et al., 2016] to introduce a more complex and nuanced

cube stacking task. The agent's goal in this environment is to stack a subset of cubes (or blocks) over a target pad in a given ordering. These cubes have distinct masses and only lighter blocks should be placed on heavier ones. Therefore, discovering temporal behaviors corresponding to sequential block stacking according to mass is beneficial. The agent is rewarded for each individual block correctly stacked according to their mass.

The task is parameterized with N_c denoting the number of cubes to be stacked. Our source task contains two blocks, i.e. $N_c = 2$, and the extrapolated transfer task contains four block, i.e. $N_c = 4$.

For behavioral cloning, we collect 17.5e3 trajectories from a scripted expert policy in the source tasks. Appendix B.3.1 provides further details on the environment and the expert policy used.

4.5.2 Method Variants and Baselines

To answer our questions above, we evaluate several variants of our proposed method, APES, together with established baselines. These variants and baselines are designed to explore the impact of hierarchy, priors, and information asymmetry on the effectiveness of skill transfer. We detail the methodological distinctions and the intended purpose of each below with appendix B.3.2 and table B.2 providing further details on the model architectures and hyperparameters used for them.

Our primary method, APES, leverages both hierarchy and learned priors to facilitate skill transfer. It also uniquely employs learned attention masks $\mathbf{m}$ to dynamically focus on relevant segments of the full observation-action history $\mathbf{x}$, optimizing the expressivity-transferability trade-off.

To investigate the importance of prior information conditioning in skill transfer (in section 4.5.3), we introduce variants of APES that employ pre-trained priors with *fixed* attention masks but differ in the extent of their information access along the temporal dimension. Ablating over the temporal dimension is a natural dimension for POMDPs where discovering belief states by history conditioning is crucial [Thrun, 1999]. APES$^{\mathrm{S}}$ conditions its high-level prior π_0^H solely on the

current state $\mathbf{s}_t$, extending the method by Galashov et al. [2019] by incorporating hierarchical structure to it. It also bears similarity with the model of Pertsch et al. [2020], with integrated low-level policies conditioned on both state $\mathbf{s}_t$ and latent $\mathbf{z}_t$. Similarly, APES$^{\text{H1}}$ aligns with the method by Bagatella et al. [2022], focusing on the immediate history, $\mathbf{x}_{t-1:t}$. APES$^{\text{H10}}$ and APES$^{\text{H20}}$ condition on longer historical data, $\mathbf{x}_{t-10:t}$ and $\mathbf{x}_{t-20:t}$, respectively. APES$^{\text{H20}}$ is akin to the approach by Tirumala et al. [2020]. These variants serve to investigate effects of information asymmetry, revealing how the balance between too little and too much information conditioning can affect performance.

To further investigate the effect of hierarchy (in section 4.5.3), we introduce two variants of APES$^{\text{H1}}$ that both enforce the regularization in the raw, rather than latent, action space. APES$^{\text{H1,FLAT}}$ denotes otherwise the same setup as APES$^{\text{H1}}$ except with a flat, non-hierarchical prior. APES$^{\text{H1,KL-A}}$ on the other hand is equivalent to APES$^{\text{H1}}$ but the regularization occurs only over the action space.

We also provide three baseline variants that learn no priors. APES$^{\text{NO-PRIOR}}$, inspired by the approach of Tirumala et al. [2019], lacks a pre-trained prior, serving as a baseline to specifically assess the contribution of learned priors to skill transfer. SAC$^{\text{REC}}$ represents a history-dependent version of SAC [Haarnoja et al., 2018d] with recurrent networks. This baseline evaluates the effectiveness of historical information without hierarchical structuring or priors. Finally, SAC$^{\text{HIER,REC}}$, the hierarchical counterpart to SAC$^{\text{REC}}$, corresponds to the model introduced by Wulfmeier et al. [2020a].

With these variants and baselines, we aim to explore the mechanisms that contribute to successful skill transfer, particularly the interplay between hierarchy, the amount of prior information conditioning, and the learned attention mechanism of APES.

4.5.3 Results

We begin our evaluations with quantitative transfer results across the outlined methods and tasks. The results are summarized in table 4.2, presenting the

Table 4.2: Average return over 100 episodes from trained final policies. The standard deviations marked with ± are computed over 7 random seeds, with the maximum attainable return in the Expert column. Experiments that do not leverage transferred priors were trained for 1e6 samples and the hierarchical ones for 1.5e5 environment steps. APES variants employ employ hierarchy and priors whereas the SAC variants omit one or the other. APES learns attention masks $\mathbf{m}$ for π_0^H whereas APES$^{\{S,H1,H10,H20\}}$ use fixed masks.

Approach	Source	π_0^H input	learn $\mathbf{m}$	Task		
				`CorridorMaze`		`CubeStack`
				$N_c = 2$ sparse interpolate	$N_c = 4$ semi-sparse extrapolate	$N_c = 4$ extrapolate
APES	-	$\mathbf{x}_{t-20:t}$	✓	$\mathbf{0.90 \pm 0.02}$	$\mathbf{6.70 \pm 0.05}$	$\mathbf{3.30 \pm 0.10}$
APES$^{\text{H20}}$	Tirumala et al. [2020]	$\mathbf{x}_{t-20:t}$	✗	0.15 ± 0.07	3.79 ± 0.14	1.10 ± 0.31
APES$^{\text{H10}}$	-	$\mathbf{x}_{t-10:t}$	✗	0.25 ± 0.07	4.12 ± 0.41	1.25 ± 0.19
APES$^{\text{H1}}$	Bagatella et al. [2022]	$\mathbf{x}_{t-1:t}$	✗	0.80 ± 0.02	6.33 ± 0.13	$\mathbf{3.13 \pm 0.09}$
APES$^{\text{S}}$	Galashov et al. [2019], Pertsch et al. [2020]	$\mathbf{s}_t$	✗	0.00 ± 0.00	2.96 ± 0.32	2.05 ± 0.11
APES$^{\text{NO-PRIOR}}$	Tirumala et al. [2019]	-	-	0.00 ± 0.00	2.30 ± 0.11	0.00 ± 0.00
SAC$^{\text{HIER,REC}}$	Wulfmeier et al. [2020a]	-	-	0.00 ± 0.00	0.07 ± 0.02	0.01 ± 0.00
SAC$^{\text{REC}}$	Haarnoja et al. [2018b]	-	-	0.00 ± 0.00	0.08 ± 0.01	0.01 ± 0.00
Expert	-	-	-	1	8	4

average returns for converged policies of both APES and SAC variants after the transfer learning phase.

Several trends emerge from the table. First, the APES variants that incorporate behavioral priors demonstrate significantly higher performance over SAC variants and APES$^{\text{NO-PRIOR}}$ that forgo the use of priors. Additionally, the performance varies significantly across APES variants with different levels of information conditioning. This influence is magnified in environments with sparse rewards, where the explicit reward signal is scarce, underscoring the necessity of deep exploration capabilities, and applies to both interpolated or extrapolated tasks. Finally, we observe that the full APES setup outperforms the other methods in each of these domains. This highlights the potential of integrated hierarchy and temporal behavioral priors together with learned attention masks.

These results also underscore the domain-dependence when configuring infor-

Table 4.3: Density of the attention masks ($\rho(\mathbf{m})$) and distillation losses ($\mathrm{D}_{\mathrm{KL}}(\pi^H \parallel \pi_0^H)$) after skill acquisition phase for `CorridorMaze` and `CubeStack` domains. Each value is the mean over 4 random seeds. The variance between seeds were negligible for all experiments.

	`CorridorMaze`		`CubeStack`	
	$\rho(\mathbf{m})$	$\mathrm{D}_{\mathrm{KL}}(\pi^H \parallel \pi_0^H)$	$\rho(\mathbf{m})$	$\mathrm{D}_{\mathrm{KL}}(\pi^H \parallel \pi_0^H)$
APES^{S}	0.04	0.81	0.03	0.75
APES^{H1}	0.05	0.22	0.05	0.65
APES^{H10}	0.50	0.12	0.50	0.49
APES^{H20}	1.00	0.11	1.00	0.47
APES	0.05	0.20	0.10	0.50
Max	1.00	0.84	1.00	1.71
Min	0.00	0.00	0.00	0.00

mation conditioning for skill transfer. Most of the methods presented in this table are analogues of existing skill transfer methods [Bagatella et al., 2022, Galashov et al., 2019, Pertsch et al., 2020, Tirumala et al., 2020]. While these methods adequately solved the tasks presented in their corresponding domains in the original works, they fail in these seemingly trivial tasks because the fixed prior conditioning does not trivially generalize across domains.

We investigate these trends in more detail below.

Information Asymmetry for Knowledge Transfer

To investigate the importance of information asymmetry for transfer, we focus on the APES variants $\text{APES}^{\{\text{S,H1,H10,H20}\}}$, which incorporate high-level behavioral priors, each with varying degree of information conditioning.

Table 4.3 presents the amount of information input to each prior policy π_0^H's (measured as the density $\rho(\mathbf{m}) := \sum_i \mathbf{m}_i / |\mathbf{m}|$, i.e. the ratio of active elements in the attention mask $\mathbf{m}$) and the distillation error (measured as the KL-divergence, $\mathrm{D}_{\mathrm{KL}}(\pi^H \parallel \pi_0^H)$) after the skill-acquisition phase. We observe an apparent linear relationship between the amount of prior information conditioning and the prior's distillation error. That is, conditioning the prior on more information improves its expressivity, thus improving its ability to distill policy's information into it, in line with theorem 4.3.2.

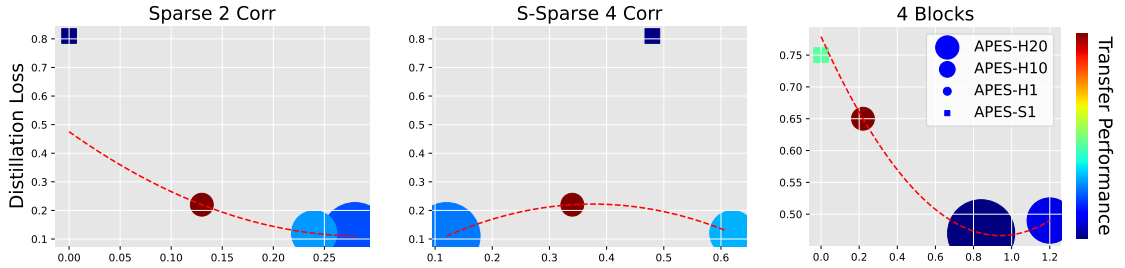


Figure 4.3: *Expressivity-Transferability Trade-Off.* Distillation loss, $\mathrm{D}_{\mathrm{KL}}(\pi^H||\pi_0^H)$, against *additional* transfer returns when reusing pre-trained π^H. The level of information conditioning is shown by marker size. Absolute transfer performance is shown by marker colour (high in red, low in blue). We fit a second-order curve, shown here as dashed red line. In sparse domains, the more information the prior π_0^H is conditioned on, the larger the benefit from reusing pre-trained π^H.

Yet, when contrasting these results with those in table 4.2, we see that the reduced distillation loss *does not* directly lead to improved transfer performance. Indeed, the data in table 4.2 indicate a trend that conditioning on either too little or too much information limits performance, hinting at an underlying *expressivity-transferability* trade-off.

The Expressivity-Transferability Trade-Off

To further investigate the performance variance observed among the APES variants, we explore whether the *expressivity-transferability* trade-off, as detailed in section 4.3 and theorems 4.3.1 and 4.3.2, dictates these outcomes.

In our default experimental setup, the high-level policy π^H is initialized randomly at the beginning of the transfer phase. However, we experiment with an alternative approach by reusing the π^H pre-trained during the skill acquisition phase. By doing so, we reduce the initial trajectory and thus the covariate shift between the source and transfer domains, as discussed in section 4.2.3. According to theorem 4.3.1, one would anticipate that priors conditioned with more information could significantly reduce covariate shift in our modified transfer scenario. And if the Covariate Shift indeed underlies the reduced transfer performance, then it stands to reason that priors with more information would derive greater benefit from utilizing a pre-trained high-level policy, as opposed to re-initializing it randomly.

This hypothesis is supported by our findings. Figure 4.3 plots the distillation loss $\mathrm{D}_{\mathrm{KL}}(\pi^H \parallel \pi_0^H)$ at the end of the skill acquisition phase against the additional transfer returns obtained by using a pre-trained π_0^H (as opposed to randomly initializing it) at the end of the transfer phase. While the effect is modest in the semi-sparse domain — likely due to its denser rewards signal offering more direct learning signal and thus mitigating covariate shift — the sparse domains demonstrate a clear trend where pre-training significantly improves the transfer performance for models conditioned with more information. Yet the final performance is still far from optimal.

These observations highlight the impact of carefully selecting the information conditioning for the prior — in essence, determining the nature and extent of information it receives — in a domain-specific manner. Optimal configuration ensures the prior only captures the task-agnostic information, thereby optimizing for improved transfer.

The discussion so far has underscored the *expressivity-transferability* trade-off governed by information asymmetry, aligning with theorems 4.3.1 and 4.3.2. Achieving the right balance in information conditioning, neither too limited nor excessive, is crucial to effective skill transfer. This segues into our next analysis, focusing on the effect of dynamically learning the attention masks to automate and optimize the selection of information asymmetry.

APES: Attentive Priors for Expressive and Transferable Skills

As seen in table 4.2, our full method, APES, outperforms the baselines and ablations on the transfer domains. Comparing APES with APES^{H20}, the most comparable approach with the prior fed the same input, $\mathbf{x}_{t-20:t}$, we observe significant performance gains.

Our results demonstrate the importance of reducing covariate shift (by minimizing information conditioning), while still supporting expressive behaviors (by exposing the prior to maximal information). As we saw in table 4.3, APES not only attends to minimal information ($\rho(\mathbf{m})$) but for that given level achieves a

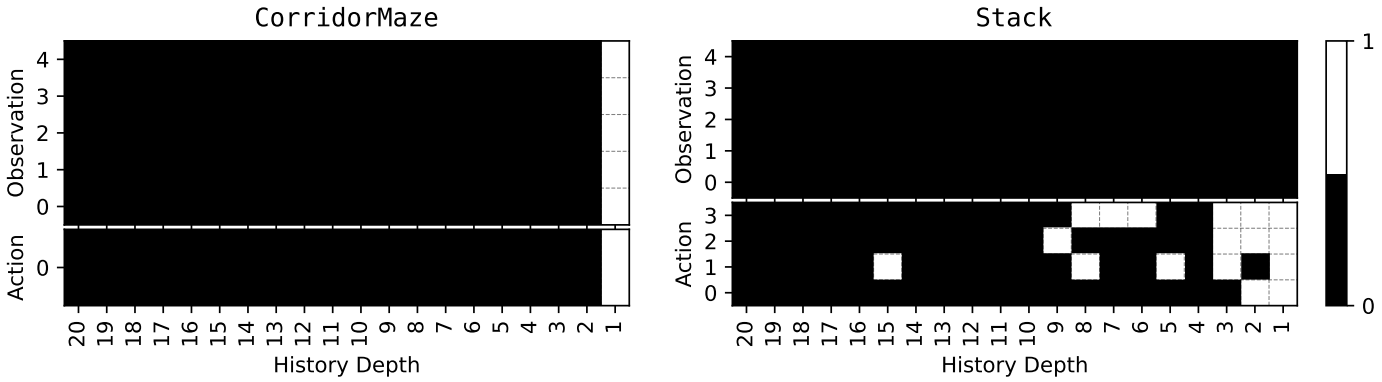


Figure 4.4: Attention masks $\mathbf{m}$ of π_0^H learned by APES, plotted for `CorridorMaze` (Left) and `CubeStack` (Right) domains. Black indicates information being masked out while white indicates information flowing through to π_0^H. The visualization highlights APES's ability to adapt its focus, yielding sparse, domain-dependent, attention patterns.

far lower distillation loss ($\mathrm{D}_{\mathrm{KL}}(\pi^H \parallel \pi_0^H)$) than methods with comparable levels information, suggesting that APES pays attention only where necessary.

Analysis of Learned Attention. To dive deeper into APES's learned attention patterns, in figure 4.4, we inspect the attention masks $\mathbf{m}_\mathbf{w}$ of π_0^H. We can observe majority of the attention values being 0, aligning APES with theorems 4.3.1 and 4.3.2.

In the `CorridorMaze` domain, the model focuses on the most recent action $\mathbf{a}_t$ and state $\mathbf{s}_t$. For a vast majority of environmental states, except when at corridor ends or intersections, the previous action is indicative enough to determine the optimal next step, i.e., to continue traversing along the corridor. For the corridor ends and intersections, information of the previous corridor is needed to act optimally. APES has learned its attention mechanism to reflect this strategy, ignoring the historical observations and actions.

In the `CubeStack` task, APES primarily allocates its attention to a short, recent history of *actions*, with the attention for actions further in the past being more sparse. Furthermore, the attention assigned to actions related to gripper manipulation (illustrated in the bottom row of figure 4.4) uses less information than the other actions. This finding reflects the functional logic of the stacking task, where successive gripper actions possess high but quickly diminishing correlation.

Table 4.4: Hierarchy Ablation in `CorridorMaze` domain. Performance averages and standard deviations over 7 seeds.

Transfer Task	`CorridorMaze` sparse, $N_c = 2$	`CorridorMaze` semi-sparse, $N_c = 4$
APES^{H1}	$\mathbf{0.80 \pm 0.02}$	$\mathbf{6.33 \pm 0.13}$
$\text{APES}^{\text{H1,KL-A}}$	$\mathbf{0.75 \pm 0.07}$	5.65 ± 0.13
$\text{APES}^{\text{H1,FLAT}}$	0.06 ± 0.03	4.42 ± 0.41
Expert	1	8

Perhaps most notably, APES masks out the observations related to the gripper's position altogether, indicating that the model effectively infers this information from other observation space aspects and recent action history. This selective attention underscores the model's capability to discern relevant from redundant information. This is interesting and aligns with the observations of Bagatella et al. [2022], demonstrating the effectiveness of *state-free* priors, conditioned on a history of actions, for effective generalization. Unlike the method of Bagatella et al. [2022], which requires domain knowledge and exhaustive hyperparameter sweeps over history lengths for effective transfer, our approach learns the length in an automated and domain-dependent manner. As such, our learned history lengths are distinct for `CorridorMaze` and `CubeStack` domains.

Hierarchy for Expressivity

To briefly investigate whether hierarchy is necessary for effective transfer in our setting, we compare APES^{H1} with $\text{APES}^{\text{H1,FLAT}}$ and $\text{APES}^{\text{H1,KL-A}}$. The transfer results for `CorridorMaze` are shown in table 4.4. Comparing $\text{APES}^{\text{H1,KL-A}}$ and $\text{APES}^{\text{H1,FLAT}}$, we see the benefits of a hierarchical prior, which is more significant for the sparse domain. Upon closer inspection, as seen in figure 4.5 and discussed in more detail below in section 4.5.3, $\text{APES}^{\text{H1,FLAT}}$ struggles with the task, primarily due to its inability to capture multi-modality at the corridor intersection. Contrasting APES^{H1} with $\text{APES}^{\text{H1,KL-A}}$, we see minimal benefits for latent regularization, suggesting that, hierarchy may not be necessary if alternative strategies for achieving multi-modality are employed.

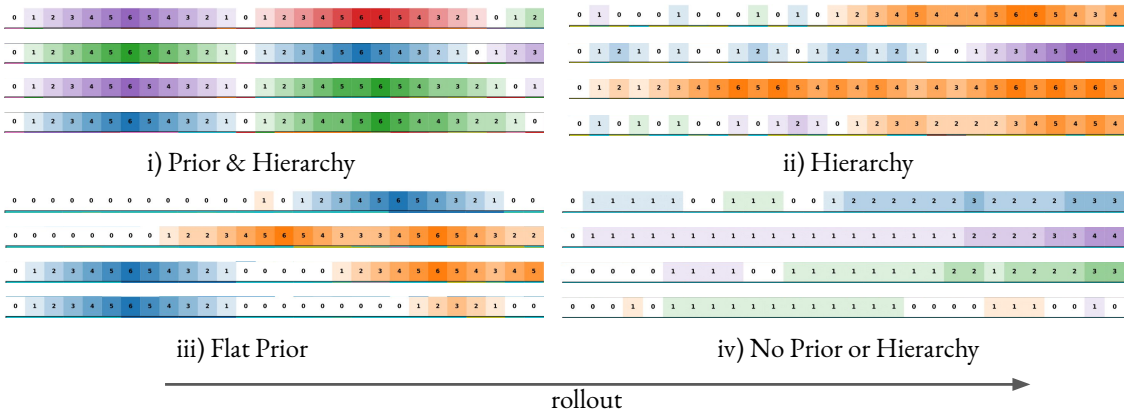


Figure 4.5: Skill-level exploration in sparse `CorridorMaze` with $N_c = 2$. 4 rollouts for each method stacked vertically, episodes unrolled horizontally. Corridors are color-coded and the agent's depth within them is denoted by shade (the darker the deeper) and number, with white 0 representing hallway, i.e. the corridor intersection. Only the full APES setup leads to corridor-level exploration.

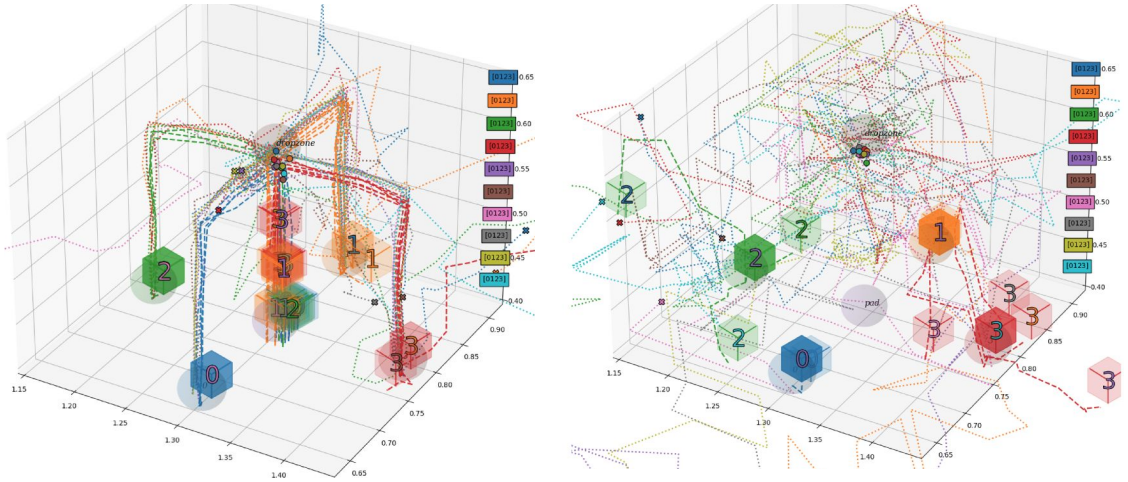


Figure 4.6: Skill-level exploration in `CubeStack` environment with number of cubes $N_c = 4$. The end-effector and block paths are depicted by dotted and dashed lines respectively, color-coded per rollout. The end positions for gripper and blocks are represented by cross and cubes, respectively. Cube numbers represent their mass order with 0 being the heaviest. APES explores at the stacking level while hierarchy alone is unable to stack. See figure B.5 for a still shot of the APES rollout frames.

Skill-Level Exploration Analysis

To gain a further qualitative understanding on the effects of hierarchy and priors, we visualize policy rollouts early on during transfer, after 5e3 steps. Within the `CorridorMaze` domain (figure 4.5), APES, utilizing both hierarchy and priors, explores effectively at the corridor level, successfully completing the sparse exploration task requiring deep exploration. In contrast, hierarchy alone, unable to express preference over high-level skills, leads to temporally uncorrelated behaviors and inability to explore different corridors. The flat prior is unable to represent multi-modal behaviors and thus explores suboptimally at the intersection of corridors, with the agent often remaining static. Without priors nor hierarchy, exploration is further hindered, rarely reaching the deeper ends of the corridors.

The trends are similar for the `CubeStack` domain, as shown in figure 4.6. APES explores at the block stacking level, stacking blocks in a sequence that respects their masses. This contrasts starkly with the setup lacking behavioral prior, which fails in its attempt to stack blocks coherently, often switching its focus prematurely and failing to satisfy the task's sequential requirements effectively.

4.6 Related Work

Hierarchical frameworks have a long history [Sutton et al., 1999b]. The options semi-MDP literature explore hierarchy and temporal abstractions [Harb et al., 2018, Igl et al., 2019, Kamat and Precup, 2020, Nachum et al., 2018, Riemer et al., 2018, Salter et al., 2022, Wulfmeier et al., 2020b]. Approaches like those by Wulfmeier et al. [2020a,b] use hierarchy to enforce knowledge transfer through shared modules. Gehring et al. [2021] use hierarchy to discover skills of varying expressivity levels. For lifelong learning [Khetarpal et al., 2020b, Parisi et al., 2019], where number of skills increase over time, it is unclear how well these approaches will fare, without priors to narrow skill exploration.

Priors have been used in various fields. In the context of offline-RL, Siegel et al. [2020], Wu et al. [2019] primarily use priors to tackle value overestimation [Levine

et al., 2020]. In the variational literature, priors have been used to guide latent-space learning [Hausman et al., 2018, Igl et al., 2019, Merel et al., 2018b, Pertsch et al., 2020]. Hausman et al. [2018] learn episodic skills, limiting their ability to transfer. Igl et al. [2019], Khetarpal et al. [2020a] learn options together with priors or interest functions, respectively. The former considers on-policy learning, limiting applicability and sample efficiency, and both of them pre-define the information conditioning of shared modules, limiting transferability. Skills and priors have been used in model-based RL to improve planning [Shi et al., 2022, Xie et al., 2020]. Sikchi et al. [2022] use priors to reduce covariate shifts during planning, while in contrast, our method ensures that the priors themselves experience reduced shifts. In the multi-task literature, priors have been used to guide exploration [Galashov et al., 2019, Pertsch et al., 2020, 2021, Siegel et al., 2020, Teh et al., 2017], yet, without hierarchy, expressivity in learned behaviors is limited. In the sequential transfer literature, priors have also been used to bias exploration [Ajay et al., 2020, Bagatella et al., 2022, Goyal et al., 2019, Liu et al., 2022, Pertsch et al., 2020, Rao et al., 2021, Singh et al., 2020], although none of these leverage hierarchy [Pertsch et al., 2020] or condition on minimal information [Ajay et al., 2020, Rao et al., 2021, Singh et al., 2020], thus limiting expressivity. Unlike APES, Bagatella et al. [2022], Singh et al. [2020] leverage flow-based transformations to achieve multi-modality. Unlike many previous works, we consider the POMDP setting, which is arguably more suited for real-world applications such as robotics, and learn the information conditioning of priors based on our *expressivity-transferability* theorems.

While most prior works rely on IA, its choice is primarily motivated by intuition. For example, Igl et al. [2019], Wulfmeier et al. [2020a,b] only employ task or goal asymmetry and Galashov et al. [2019], Merel et al. [2020], Tirumala et al. [2019] use exteroceptive asymmetry. Salter et al. [2020] investigate a way of learning asymmetry for sim2real domain adaptation, but condition $\mathbf{m}$ only on observation and state. We provide a more principled investigation on the role of IA for transfer, proposing a method for automating this choice.

4.7 Conclusion

In this chapter, we explored hierarchical KL-regularized RL to efficiently transfer skills across sequential tasks, showing the effectiveness of combining hierarchy and priors. We demonstrated the crucial *expressivity-transferability* trade-off in skill transfer, which is controlled by the choice of information asymmetry. Our experiments validate the importance of this trade-off for both interpolated and extrapolated domains. To address this trade-off, we introduced *A*ttentive *P*riors for *E*xpressive and Transferable *S*kills (APES), a novel method designed to automate the selection of information asymmetry for the high-level prior by learning it in a domain-specific and data-driven manner. This is achieved by feeding the entire history to the prior, capturing *expressive* behaviors, while encouraging its attention mask to be sparse, thus minimizing covariate shift and improving *transferability*. Our experiments across domains of varying sparsity levels demonstrate how APES consistently outperformed existing methods, while bypassing arduous information-asymmetry sweeps. Furthermore, our ablations demonstrated the importance of hierarchy for prior expressivity by supporting multi-modal behaviors.

One of the key challenges in expanding the real-world applicability of APES is the availability and diversity of source tasks, as well as the data-generating policies needed for them. In the next chapter, we shift our focus to more complex full-body humanoid agents. While more challenging to control, such embodiments can benefit from the plethora of example motions available from motion capture databases and animated keyframe motions. Similar skill-learning methods as we presented in this chapter have been successfully demonstrated for learning full-body controllers to reuse the learned motor primitive modules to solve more complicated downstream tasks. We investigate a method for obtaining expert demonstrations in such a setting.

5

Interactive Full-Body Motion Imitation with Model-Predictive Control

Contents

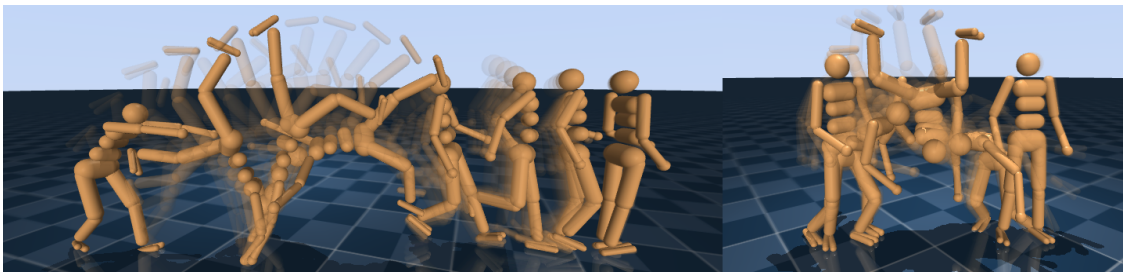


Figure 5.1: Our MPC-based framework enables high-quality physics-based motion tracking at interactive speeds on a consumer-grade laptop. `SimpleHumanoid` performing cartwheel (left) and front flip (right).

5.1 Introduction

The ability to synthesize natural and meaningful behaviors has been a longstanding goal for character animation and robotics. Physics-based controllers, together with large motion capture databases and animated keyframe motions, serve as powerful priors and have emerged as a promising approach for this purpose Bohez et al. [2022], Chentanez et al. [2018], Fussell et al. [2021], Hasenclever et al. [2020], Lee et al. [2010], Liu et al. [2010, 2016], Luo et al. [2021, 2023a], Merel et al. [2017, 2018b], Muico et al. [2009], Peng et al. [2017, 2018a, 2022], Wagener et al. [2022], Wang et al. [2020], Won et al. [2020], Yin et al. [2007], Yuan and Kitani [2019].

The recent literature has predominantly focused on reinforcement learning (RL) methods to train motion tracking controllers Bohez et al. [2022], Chentanez et al. [2018], Fussell et al. [2021], Hasenclever et al. [2020], Luo et al. [2021, 2023a], Merel et al. [2017, 2018b], Peng et al. [2017, 2018a, 2022], Wagener et al. [2022], Wang et al. [2020], Won et al. [2020], Yuan and Kitani [2019]. Despite their ability to produce an impressive range of motions, from basic movements to complex acrobatics, these methods are often challenging to apply in practice. The intricacies involved in hyperparameter tuning and task design, combined with long training times and slow feedback loops, render these approaches laborious and prone to errors. Furthermore, the current state-of-the-art RL-based universal tracking policies are still unable to reliably reproduce motions beyond the training datasets.

Concurrently, trajectory optimization methods, once a staple for motion tracking, have seen a decline in usage, perhaps owing to their complex implementation and limited accessibility for non-expert users. Inspired by the recent advances in simulation and model-based planning software Howell et al. [2022], Todorov et al. [2012], we revisit physics-based motion tracking with model predictive control (MPC). In contrast to the learning-based strategies, MPC methods obviate the need for an offline training phase, thereby offering animators and robotic task designers faster feedback loops and increased productivity. Our implementation, built upon the MuJoCo MPC framework Howell et al. [2022], is extremely simple

and enables near-real-time tracking on standard consumer-grade laptops, facilitating user interaction with the system through MuJoCo MPC's graphical user interface.

We argue for a more comprehensive evaluation of motion tracking techniques that includes trajectory optimization methods alongside RL-based models. Our study compares the MPC-based approach with recent state-of-the-art RL-based universal tracking controllers Luo et al. [2023a], demonstrating superior tracking quality, particularly for sequences not included in the training dataset. This comparison underscores the potential for improvement in RL-based motion tracking and shows that trajectory optimization methods serve as a robust alternative to RL approaches.

Our main contributions are two-fold. Firstly, we introduce an interactive MPC-based motion tracking system built upon the MuJoCo MPC Howell et al. [2022] framework, marking, to our knowledge, the first open-source implementation of its kind. Secondly, we present a comprehensive quantitative analysis comparing our MPC method with a state-of-the-art RL-based motion tracking controller on a large-scale motion dataset. Our findings establish a new baseline for motion tracking performance and highlight the effectiveness of MPC in scenarios where current RL-based solutions still fall short.

5.2 Related Work

Character motion synthesis has a long history spanning various fields, including biomechanics, robotics, and computer graphics. This section presents a short overview of the most closely related works in physics-based motion tracking and online behavior synthesis.

5.2.1 Kinematic Motion Generation

The task of kinematic motion generation, where 3D keyframe motions are generated without imposing hard physics constraints, has a long history in character animation. Early research predominantly focused on human motion analysis and prediction, leveraging statistical models to anticipate future frames based on past motion or initial poses Badler et al. [1993], Gavrila [1999], O'Rourke and Badler [1980]. These

models have gradually advanced, with recent studies incorporating neural network-based generative models like GANs and VAEs for future motion prediction Barsoum et al. [2018], Komura et al. [2017]. This shift has allowed for more intricate and detailed kinematic motion generation, such as those diversifying the motion distribution Yuan and Kitani [2020] and focusing on motion in-betweening or blending Duan et al. [2021], Harvey and Pal [2018], Harvey et al. [2020], Jiang et al. [2023], Kaufmann et al. [2020].

Today's rich motion databases have enabled kinematic motion generation models to introduce novel input modalities for guiding the motion generation. For instance, recent methods have successfully generated dance motions synchronized with musical inputs Lee et al. [2019], Li et al. [2020], Tseng et al. [2023]. Various methods have also been proposed for the synthesis of motions conditioned on discrete motion categories Ahn et al. [2018], Ahuja and Morency [2019], Dilokthanakul et al. [2016], Guo et al. [2020], Lin and Amer [2018], Maheshwari et al. [2022], Petrovich et al. [2021]. Finally, the latest methods are capable of solving several motion synthesis tasks, including generating motions from free-form natural language instructions or composing styles, all within a singular model Jiang et al. [2023], Kim et al. [2023], Tevet et al. [2022a,b], Zhang et al. [2022].

5.2.2 Physics-Based Motion Tracking

Incorporating physics into the motion synthesis loop Bergamin et al. [2019], Chen et al. [2019], Fu et al. [2023], Fussell et al. [2021], Gong et al. [2022], Hasenclever et al. [2020], Makoviychuk et al. [2021], Park et al. [2018, 2019], Peng et al. [2017, 2018a, 2019a], Wagener et al. [2022], Yuan and Kitani [2018] provides useful prior knowledge of how the environment evolves, lifting the burden of modeling object interactions or other external forces from the designer to the simulator. Furthermore, applications in robotics require physical realism and motor-actuated motions, which are not accommodated by kinematic methods.

Reinforcement Learning Methods. In recent years, reinforcement learning has drawn significant interest from the physics-based motion synthesis community. Various methods produce policies capable of imitating a handful of reference motions. Among these, Merel et al. [2018a], Park et al. [2019], Peng et al. [2018a], Ren et al. [2023] train policies conditioned on a phase-variable and optimize the tracking objective using reinforcement learning, while others, such as Merel et al. [2017], Peng et al. [2018b, 2021], Wang et al. [2017] train an adversarial motion discriminator and use it as an imitation reward for the reinforcement learner. Many of these methods also facilitate additional user control either through task objectives Peng et al. [2018a, 2021] or through hierarchical latent space models where the tracking policy serves as the low-level controller for a downstream task Bohez et al. [2022], Fussell et al. [2021], Hasenclever et al. [2020], Luo et al. [2023b], Merel et al. [2018a,b, 2020], Peng et al. [2022]. Our work focuses on motion tracking but could similarly be extended to allow user control.

Works most related to ours are those that learn universal tracking controllers. The goal of these models is to be able to faithfully track arbitrary reference motions with a single parameterized model, thus enabling interactive, online motion synthesis by tracking user-provided keyframe sequences without necessitating a separate offline training phase for each new motion. Hasenclever et al. [2020], Merel et al. [2018b], Wagener et al. [2022] all distill expert policies into an encoder-decoder based architecture, reporting 70-85% performance relative to the expert policies on the CMU MoCap database CMU Graphics Lab [2003]. Won et al. [2020] train a mixture-of-experts policy with similar performance. Chentanez et al. [2018], Wang et al. [2020] similarly track larger motion databases with a single policy but do not provide comparable quantitative results of their tracking performance. More recent methods use the even larger AMASS dataset Mahmood et al. [2019]. Luo et al. [2021] successfully imitates 97% of the AMASS dataset in a simplified setting where physically unrealistic "magic" forces, originally introduced by Yuan and Kitani [2019], are applied at the root of the humanoid pulling it towards the target motion direction. PHC by Luo et al. [2023a] lifts this simplification and

focuses on improving failure recovery by introducing difficulty-based curriculum to the training regime, achieving performance on par with UHC. Fussell et al. [2021] approach motion tracking from the perspective of model-based reinforcement learning. Their method, SuperTrack, uses model-based reinforcement learning and notably accelerates training speed relative to the more widely used model-free, on-policy reinforcement learning approaches. Despite their impressive qualitative results, their quantitative results demonstrate limitations when applied to motions outside the LAFAN dataset Harvey et al. [2020], on which the model was trained.

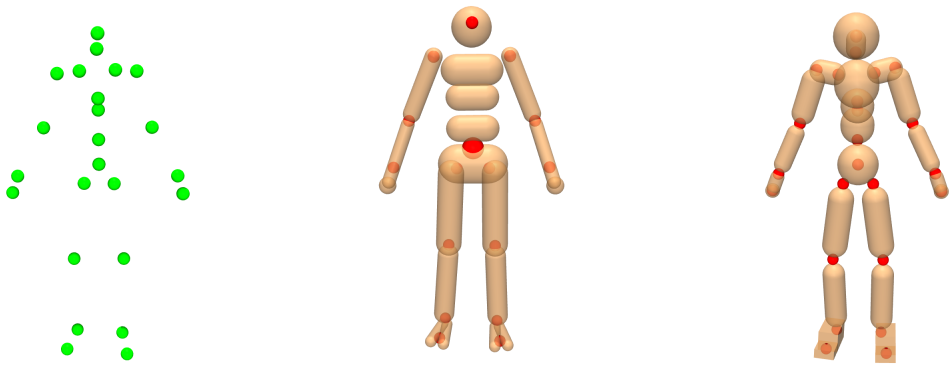
The diversity of reported evaluation metrics makes direct comparison of these methods difficult and underscores the lack of standardization in the field. Our work contributes to this discussion by pointing out shortcomings with the currently-used success metrics and proposing a stricter yet simple alternative.

Trajectory Optimization Methods. Model-based methods utilizing reference motions have a similarly long history in character motion synthesis. SIMBICON by Yin et al. [2007] uses a data-driven balancing mechanism for bipedal characters and couples it with feedback error learning to track simple cyclic walking motions. da Silva et al. [2008] introduced a similar system, replacing feedback error learning with quadratic programming, for walking controllers that exhibit improved robustness and stability. Da Silva et al. [2008] further improved the method with LQR to enable non-cyclic motions such as standing, stepping, and transitions between the two. Muico et al. [2009] employed an even more sophisticated Nonlinear Quadratic Regulator controller for full-body humanoid tracking. Lee et al. [2010] follow SIMBICON-like balance strategies and use inverse dynamics to track various styles of 3D locomotion data, which can be delivered on-the-fly without the need for pre-processing. These methods are all limited in the range of motions they are able to imitate, rely on hand-crafted heuristics, or require an offline training phase. Our method does not require any offline learning phase while also being capable of reliably tracking a much broader range of dynamic motions.

Another line of work Liu et al. [2010, 2015] solves the motion tracking problem using sampling-based offline planning. They are capable of imitating a wide array of behaviors, such as dynamic and acrobatic motions, yet their system is relatively complex and the controller relies heavily on low-dimensional state representation. Additionally, the sampling strategy is slow, requiring long offline planning, thus being unsuitable for interactive applications. Hämäläinen et al. [2014, 2015] show that improved sampling-based strategies are capable of synthesizing simple motions such as getting up, standing, and balancing under external perturbations with hand-crafted objectives, without motion tracking. In a similar fashion, Tassa et al. [2012] synthesize simple humanoid motions at interactive rates with manually-designed objectives, without reference motions, using iLQG. Our work focuses on interactive tracking.

The method closest to ours is that of Han et al. [2016], who similarly use iLQG for interactive motion tracking. Their main contribution is to improve the speed of the iLQG planner. Our work uses a simpler version of iLQG and focuses on investigating the tracking quality. While the tracking quality of Han et al. [2016] is impressive, they only test their system on a handful of motions and provide no quantitative results on the tracking performance. We provide numbers for thousands of motion clips, establishing a strong baseline for RL-based methods.

Robotics Various works have also leveraged motion capture data and MPC for robotic control. From the abovementioned methods, Bohez et al. [2022] transfer a tracking policy learned in simulation to a real-world quadruped, enabling it to perform behaviors like walking and dribbling a ball. Escontrela et al. [2022] build upon the work by Peng et al. [2021] to develop policies that transfer effectively from simulated to a real quadrupedal robot, by utilizing style rewards learned from reference motions. Reske et al. [2021] distills plans generated by MPC into a parametric policy network that can execute different gaits in real-time and effectively replace the MPC on hardware. While these methods yield basic locomotion skills, such as trotting and static walking, our work focuses on simulated



(a) AMASS keyframe. **(b)** `SimpleHumanoid`. $N_J = 16$. **(c)** `SMPLHumanoid`. $N_J = 24$.

Figure 5.2: Simulated humanoids and an example keyframe pose. Our MPC cost operates directly in the Cartesian 3D space, thus bypassing the inverse kinematics steps required in the retargeting phase used by prior methods.

character animation, which not only enables but also demands more dynamic and visually compelling motions.

5.3 Background

Motion Tracking. Our goal is to control a motor-actuated character so as to faithfully track given reference motions. The character model is structured as a tree of rigid body elements, each representing a segment of the character. This tree structure is interconnected by motor-actuated joints, allowing for articulated movement. At any given moment, the state of the character is defined as $\mathbf{x} = [\mathbf{q}, \mathbf{v}, \mathbf{a}]$, where $\mathbf{q}$ represents the joint positions, $\mathbf{v}$ the joint velocities, and $\mathbf{a}$ the actuator activation states. The control over the character's movements is achieved by applying actuation commands $\mathbf{u}$ to the joints. In addition to $\mathbf{x}$, the controller receives as an input a predefined kinematic reference motion, which may be obtained from various sources: motion capture systems that record real-life movements, sequences hand-authored by keyframe animators, or dynamically generated from pre-trained keyframe motion generation models. We denote the reference *pose* that the character aims to imitate by $\hat{\mathbf{p}} \in \mathbb{R}^{N_J \times 3}$, which consists of the Cartesian positions of the N_J sites of interest. An example of such a pose is visualized in figure 5.2a. The sequence of these poses over time forms the complete reference *motion* (also referred

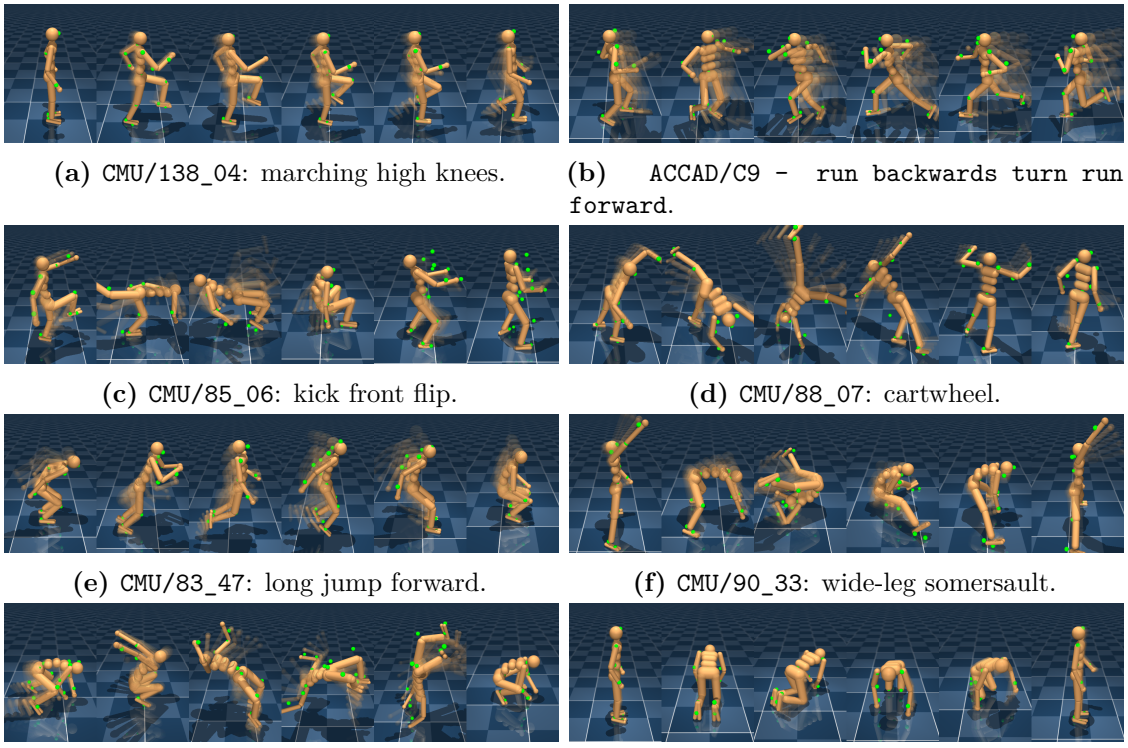
to as a *trajectory* or *sequence* from here on) and is denoted as a temporally evenly spaced sequence of N_T poses $\hat{\mathbf{m}} \in \mathbb{R}^{N_T \times N_J \times 3}$. Given a reference motion, our system uses model-predictive control to synthesize the motor actuations that enable the character to reproduce that behavior.

Motion Retargeting. The reference motions seldom align perfectly with the simulated character's morphology. To address this discrepancy, we select a subset of the keypoints from the reference motion and pair them with manually positioned sites on the character's body, as illustrated in figures 5.2b and 5.2c. For our humanoid, these sites may include, for example, the pelvis, torso, hips, knees, ankles, toes, shoulders, elbows, and wrists. During simulation, the Cartesian site positions, denoted by $\mathbf{p} \in \mathbb{R}^{N_J \times 3}$ (with a slight abuse of notation, we repurpose N_J to denote the number of selected keypoints), are obtained via forward kinematics as a function of $\mathbf{q}$. Analogously to the reference motion, the materialized character motion is defined as a temporally evenly spaced sequence of N_T poses, $\mathbf{m} \in \mathbb{R}^{N_T \times N_J \times 3}$.

Optimal Control. We formulate the motion tracking task as an optimal control problem with a finite-horizon Markov Decision Process (MDP) defined by a tuple $(\mathcal{X}, \mathcal{U}, f, c, H)$. The state space $\mathcal{X}$ and action space $\mathcal{U}$ are assumed to be continuous, and the environment evolves according to the discrete-time dynamics $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$, that is, the environment transitions to the next state $\mathbf{x}_{t+1} \in \mathcal{X}$ given the current state $\mathbf{x}_t \in \mathcal{X}$ and an action $\mathbf{u}_t \in \mathcal{U}$, with subscripts t denoting discrete time steps. The environment emits a cost $c(\mathbf{x}_t, \mathbf{u}_t) : \mathcal{X} \times \mathcal{U} \to [0, \infty]$ on each transition.

We focus on the *finite-horizon* formulation, whereby our objective is to minimize the cumulative sum of costs starting from state $\mathbf{x}$:

$$\begin{aligned} J(\mathbf{u}_{1:H}) &= \sum_{t=1}^{H} c(\mathbf{x}_t, \mathbf{u}_t), \\ \text{subject to} \quad \mathbf{x}_1 &= \mathbf{x}, \\ \mathbf{x}_{t+1} &= f(\mathbf{x}_t, \mathbf{u}_t). \end{aligned} \tag{5.1}$$



(a) `CMU/138_04`: marching high knees.

(b) `ACCAD/C9 - run backwards turn run forward`.

(c) `CMU/85_06`: kick front flip.

(d) `CMU/88_07`: cartwheel.

(e) `CMU/83_47`: long jump forward.

(f) `CMU/90_33`: wide-leg somersault.

(g) `CMU/90_19`: monkey backflip.

(h) `BMLmovi/Subject_17_F_5`: crawl.

Figure 5.3: Motion synthesized with the MJPC tracking controller. Sites from reference motions are denoted by the green spheres.

5.4 Method

Trajectory Optimization with iLQG. There are various methods for solving the trajectory optimization problem in equation (5.1). We choose to use the iterative Linear Quadratic Gaussian (iLQG) planner[1], due to its favorable characteristics for motion tracking tasks. The dense nature of the tracking objective reduces the need for deeper exploration, making a gradient-based method like iLQG well-suited for the task.

iLQG is a Gauss-Newton approximation of the traditional Differential Dynamic Programming (DDP) algorithm Jacobson and Mayne [1970], with the key distinction being its reliance only on the first derivatives of dynamics, as opposed to using second derivatives like in DDP. As a result, iLQG loses the quadratic convergence

[1]As in Howell et al. [2022], our implementation actually uses iLQR but we choose to use the name iLQG due to its provenance, even though we don't make use of the noise-sensitive term, for which iLQG was originally developed for Li and Todorov [2004].

characteristic provided by DDP. However, within the context of MPC, where the optimum continuously shifts and convergence is not typically expected, the advantage of faster evaluation provided by iLQG often significantly outweighs the reduction in performance.

The details of iLQG are too involved to be detailed here and we refer interested readers to Tassa et al. [2012] for the details. In summary, iLQG exploits the recursive structure of the optimization problem to efficiently produce a time-varying linear feedback policy:

$$\mathbf{u}_t = \bar{\mathbf{u}}_t + \alpha k_t^* + K_t^*(\mathbf{x}_t - \bar{\mathbf{x}}_t) \tag{5.2}$$

where the overbars $\bar{\mathbf{x}}$ and $\bar{\mathbf{u}}$ denote the state and control values from the *nominal*, or current best trajectory, k^* is the improvement to the current action trajectory, and K^* is the feedback gain matrix. As in Tassa et al. [2012], $0 < \alpha <= 1$ denotes the backtracking line-search parameter, which is iteratively adjusted to find an optimal improvement. For more in-depth information on the line-search procedure and backward pass regularization, we refer the reader to Tassa et al. [2012].

In practice, our Model Predictive Control system consists of two separate processes: the *agent* and the *planner*. The iLQG planner continuously improves its policy at time step t with the corresponding state $\mathbf{x}_t$, obtained from the agent. The agent advances its simulation at time step t by executing the first action, $\mathbf{u}_t$, from the most up-to-date nominal from the planner. As discussed in Howell et al. [2022], Tassa et al. [2012], these processes can be run asynchronously and the agent simulation can be slowed down to provide the planner with more planning time, thus improving the policy's performance.

Control Space. Our system uses stateful, filtered position actuators at each of the character's joints, resulting in 3rd-order actuation dynamics. The policy's control signal $\mathbf{u}$ specifies the position target for the joint, which is translated into the actuator's internal activation state, which then determines the output force based on an affine function of this state and the actuator's dynamic properties.

Specifically, the force generated by each actuator, $\mathbf{f}_i$ applied at each joint i is derived from the activation state $\mathbf{a}$, as $\mathbf{f} = \theta^{\text{gain}} \cdot \mathbf{a} + \theta_0^{\text{bias}} + \theta_1^{\text{bias}} \cdot \mathbf{l}$, where θ^{gain} represents the actuator's sensitivity to the activation state, θ_0^{bias} is a constant bias, θ_1^{bias} adjusts the force based on the actuator's length, and $\mathbf{l}$ denotes the length of the actuator. The activation dynamics are defined as $\dot{\mathbf{a}} = (\mathbf{u} - \mathbf{a})/\theta^{\text{dynamics}}$, where θ^{dynamics} is a constant parameter specifying the actuator's response speed. We found such an actuator model to produce smoother and less jittery behaviors compared to the commonly used PD or direct torque control used by the prior works Luo et al. [2023a,b], Peng et al. [2018a], particularly when combined with the activation regularization as described below. In the following, we assume that the actuator states $\mathbf{a}$ and $\dot{\mathbf{a}}$ are included in the state $\mathbf{x}$.

Tracking Cost. A critical factor in tracking performance is the design of the cost function, c. Utilizing an interactive MPC framework enables any parameter change to be immediately reflected in the performance of the controller, making the design process extremely fast. This contrasts with the policy optimization methods like reinforcement learning that necessitate a long offline training phase to observe changes in performance due to modifications in the cost function.

For brevity, we introduce shorthand notations for tracking errors and norms used in the final cost. Let $\delta \in \mathbb{R}^{N_T \times N_J \times 3}$ represent the element-wise tracking error, where each element $\delta_{t,j,i}$ indicates the absolute difference between the actual and reference positions of site j at time step t: $\delta_{t,j,i} = |\mathbf{m}_{t,j,i} - \hat{\mathbf{m}}_{t,j,i}|$. We also use a modified L2 norm, $n_\alpha^{L2}(x) = \sqrt{x^T x + \alpha^2} - \alpha$, which behaves smoothly in an α-sized neighborhood around the origin and approaches the regular L2 distance further away. The norm is simple, providing closed-form first- and second-order derivatives, thus allowing for fast Gauss-Newton Hessian approximation for iLQG.

We arrive at the following running tracking cost:

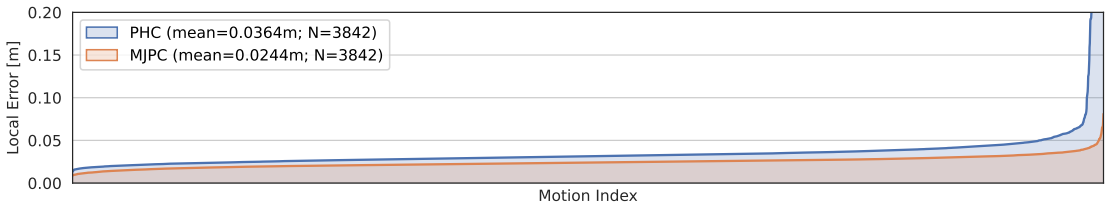
$$\begin{aligned} c(\mathbf{x}_t, \mathbf{u}_t) &= \mathbf{w}^{\mathbf{s}} c^s(\mathbf{x}_t) + w^u c^u(\mathbf{u}_t) + w^a c^a(\dot{\mathbf{a}}_t), \\ \text{where} \quad c^s(\mathbf{x}_t) &= \left[\, n^{L2}_{\alpha^s_0}(\delta_{t,0}) \, , \, n^{L2}_{\alpha^s_1}(\delta_{t,1}) \, , \, \ldots \, , \, n^{L2}_{\alpha^s_{NJ}}(\delta_{t,NJ}) \, \right], \\ c^u(\mathbf{u}_t) &= n^{L2}_{\alpha^u}(\mathbf{u}_t), \\ c^a(\dot{\mathbf{a}}_t) &= n^{L2}_{\alpha^a}(\dot{\mathbf{a}}_t). \end{aligned} \tag{5.3}$$

The $c^s(\mathbf{x}_t)$ component in the cost discourages the tracking sites from deviating from their corresponding reference positions, while $c^u(\mathbf{u}_t)$ and $c^a(\dot{\mathbf{a}}_t)$ regularize the actions and the actuator activations. We find using filtered position actuators together with the actuator activation regularization particularly effective for producing smoother motions and reducing jitter in the motions.

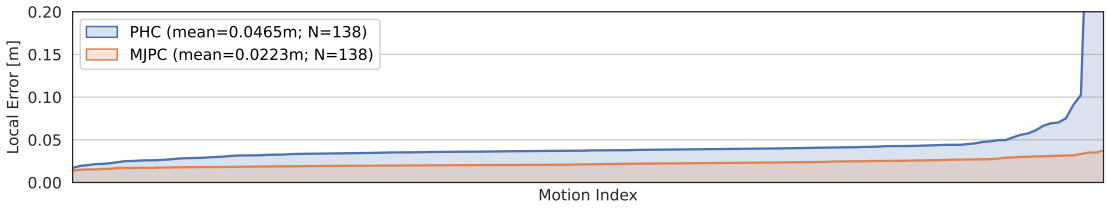
Our cost differs from those used in related motion tracking methods Hasenclever et al. [2020], Luo et al. [2023a], Merel et al. [2018b], Peng et al. [2018a] in two ways. First, we omit the often-used velocity components from our objective, thus simplifying the cost without compromising on the quality of motion tracking. In our experiments, the position cost together with simple regularization was enough to produce high-quality tracking.

Second, our method operates on Cartesian site position errors rather than joint position discrepancies. Obtaining the joint positions $\hat{\mathbf{q}}$ from the reference poses $\hat{\mathbf{p}}$ often requires an additional inverse kinematics step Hasenclever et al. [2020], Luo et al. [2023b], Merel et al. [2018b], Peng et al. [2018a]. The commonly used inverse kinematics methods process each frame in the sequence independently and thus disregard the temporal structure of the motion. This myopic behavior can result in $\hat{\mathbf{q}}$ values that contain abrupt and unnatural jumps for motions that necessitate extreme joint positions. By operating directly on the Cartesian 3D poses, we bypass the inverse kinematics steps, and our method can be seen as implicitly doing physics-guided inverse kinematics.

Episode Initialization. While our tracking objective only requires the Cartesian reference positions $\mathbf{p}$, initializing each tracking episode still requires the initial joint configuration $\mathbf{q}_0$ and $\mathbf{v}_0$. We simply follow prior work and obtain the joint positions



(a) $E^{\text{mpjpe-l}}$ for 3842 randomly chosen sequences from AMASS train split.



(b) $E^{\text{mpjpe-l}}$ for the 138 AMASS test sequences.

Figure 5.4: Per-sequence tracking errors for the keyframe motions from AMASS splits from Luo et al. [2023a]. Each point on the x-axis presents a motion in the dataset and the y-axis presents local tracking error, $E^{\text{mpjpe-l}}$ (lower is better), for the training set (a, top) and the test set (b, bottom). For better readability, the motions within each method are shown in ascending order based on the corresponding tracking error. The average error over the motions for each split is shown in the corresponding legend.

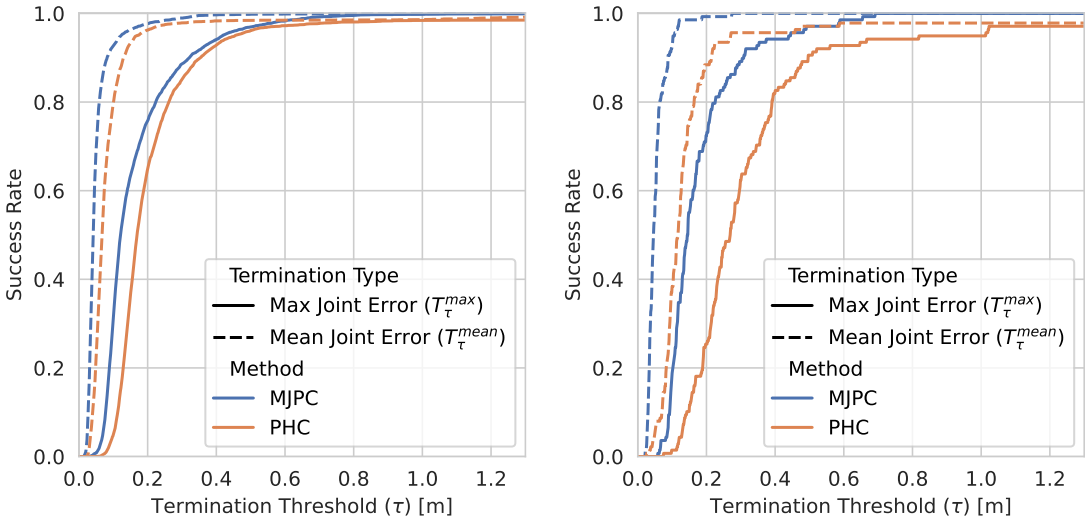
$\mathbf{q}_0$ through inverse kinematics Buss [2004] from the initial site positions $\hat{\mathbf{p}}_0$, and $\mathbf{v}_0$ as the finite difference between the first two joint positions $\mathbf{q}_{0:1}$.

5.5 Experiments

We assess our MPC-based framework, from here on referred to as MJPC, across a large variety of humanoid motions. Our objective is to evaluate MJPC's tracking performance on a diverse set of reference motions and compare it with that of reinforcement learning-based tracking methods. We demonstrate that MJPC outperforms RL-based methods, particularly for motions outside the RL policy's training domain.

5.5.1 Experimental Setup

Software. Our simulations use the MuJoCo physics engine Todorov et al. [2012] and the MPC controllers implemented on the MuJoCo MPC (MJPC) framework Howell et al. [2022]. Reset-state inverse kinematics is performed with the high-level functions from the DeepMind Control Suite Tunyasuvunakool et al. [2020].



(a) Subset of AMASS train set. N=3842.

(b) AMASS test. N=138.

Figure 5.5: Pareto charts of success rate versus termination threshold τ for keyframe motions from AMASS train and test splits introduced in Luo et al. [2021, 2023a]. In addition to the success based on mean joint error, T_τ^{mean}, as reported in Luo et al. [2021, 2023a], we also include the maximum joint error, T_τ^{max}.

The results for the RL baseline are obtained using the authors' official open-source implementation that uses the Isaac Gym simulator.

Characters. Our experiments feature two humanoid models: the 21-Degree of Freedom (DoF) `SimpleHumanoid` from DeepMind Control Suite as introduced by Tunyasuvunakool et al. [2020] and a modified version of the 69-DoF `SMPLHumanoid` by Luo et al. [2023a]. While Luo et al. [2023a] provide a MuJoCo-compatible model of their `SMPLHumanoid`, we find it to be unstable out of the box. We attribute this instability to the model's original design targeting the Isaac Gym simulator. To address this, we modify the `SMPLHumanoid`'s joints and actuators. We provide comprehensive details of both the original and modified model parameters in appendix C.2 for further reference.

Datasets. We use AMASS, a large motion capture dataset introduced by Mahmood et al. [2019], including more than 40 hours of motions compiled from multiple public motion databases. While generally high quality, some motions in AMASS contain physically implausible behaviors such as object interactions and unrealistic

body penetrations. We adopt the dataset splits originally introduced by Luo et al. [2021] and subsequently curated by Luo et al. [2023a], which largely omits the physically infeasible sequences and results in final splits of 11313 training sequences and 138 test sequences.

Baselines. We compare MJPC with the state-of-the-art motion tracking model, PHC, introduced by Luo et al. [2023a]. To ensure a fair and accurate comparison, all results are obtained using the authors' official open-source implementation. The authors report tracking performance both for a model trained with a joint position-based objective and a model trained with Cartesian 3D site positions. In the following, we only report numbers for their joint position-based model, which generally performs better than their site position-based model.

Model and Planner Parameters. The parameters affecting the tracking quality the most are simulation model's time step, that is, the time elapsed between two discrete simulation steps, and the planner horizon, H. We find the maximum stable time step by incrementally increasing its value in the GUI until the simulation exhibited instability. We end up with $1/60$ for `SimpleHumanoid` and $1/120$ for `SMPLHumanoid` and use the same time step for both agent and planner. To select the planning horizon, we tested a series of motions characterized by extended flight phases, gradually increasing the horizon until the planner performs well. The final value used for both models is $0.45s$ horizon corresponding $H = 27$ for `SimpleHumanoid` and $H = 54$ for `SMPLHumanoid`. The rest of the hyperparameters, including the joint and actuators parameters, are detailed in appendix C.3.

Hardware and Planning Speed. All our experiments and development process run efficiently on CPU-only hardware, thus eliminating the need for specialized accelerators such as GPUs. A consumer-grade Apple MacBook Air M2 is capable of executing the majority of `SimpleHumanoid` motions at 30% of real-time speed, and more demanding ones, such as cartwheels or contact-rich motions, at 20% real-time speed. This performance is achieved through the use of asynchronous

planning and the graphical user interface provided by MuJoCo MPC. For more computationally demanding experiments, such as the `SMPLHumanoid` comparisons with PHC, we use a CPU server with 4 CPU cores allocated for each MJPC worker (i.e. per each motion), and run the planner synchronously. Due to its higher degrees of freedom and comparatively lower model quality, which necessitates smaller simulation time steps, the `SMPLHumanoid`'s execution speed is slower, running at approximately 1% of real-time speed.

Metrics. We assess tracking quality using two error metrics introduced by Luo et al. [2021, 2023a]: global mean per-joint position error (MPJPE), denoted as $E^{\text{mpjpe-g}} = \frac{1}{N_T N_J} \sum_{t,j} \delta_{t,j}^{\text{L2}}$, and root-relative MPJPE, denoted as $E^{\text{mpjpe-l}}$, both measured in meters. We adopt two distinct definitions for failures:

$$T_\tau^{\text{mean}}(\mathbf{p}_t, \hat{\mathbf{p}}_t) = \frac{1}{N_J} \sum_j \delta_{t,j}^{\text{L2}} \; > \tau \tag{5.4}$$

$$T_\tau^{\text{max}}(\mathbf{p}_t, \hat{\mathbf{p}}_t) = \max_j \delta_{t,j}^{\text{L2}} \quad > \tau, \tag{5.5}$$

where τ[2] denotes a termination threshold and $\delta_{t,j}^{\text{L2}} = \|\mathbf{m}_{t,j} - \hat{\mathbf{m}}_{t,j}\|_2$ the standard L2-error for site j at time step t. The criterion T_τ^{mean} follows the failure definition of Luo et al. [2021, 2023a], deeming tracking unsuccessful if the average deviation of body joints from the reference motion exceeds τ meters at any point during the sequence. Conversely, T_τ^{max}, defines the tracking unsuccessful if *any* joint at any point deviates more than τ meters from its target.

5.5.2 Results

Qualitative Evaluation. We first showcase motions synthesized by MJPC in figure 5.3. Our method is able to faithfully track a wide range of motions, from simple walking and running motions (figures 5.3a and 5.3b) to more dynamic jumps (figure 5.3e) and flips (figures 5.3c and 5.3g) requiring long flight phases, as well as contact-rich motions, such as crawling and rolling on the floor (figure 5.3h).

[2]Note the reuse of notation from previous chapters.

Tracking Error. We then provide a comparative analysis of MJPC and PHC motion tracking results on the AMASS train and test sets in figure 5.4. When evaluated through scalar metrics, as in Luo et al. [2021, 2023a], MJPC demonstrates superior performance over PHC in both local (figure 5.4) and global tracking errors (figure C.1, appendix C.1). MJPC's performance is consistent across the motions within each dataset, whereas PHC shows greater variability between motions. Qualitatively, we observe that for motions with small errors, both MJPC and PHC produce visually satisfactory results. PHC's slightly higher error rates in these instances might be attributed to its motion prior objective, which effectively trades off minor inaccuracies for the naturalness of motion. However, PHC encounters massive tracking errors in several motions, in contrast to MJPC, which maintains consistent performance across the datasets. Upon visual inspection, PHC's failures appear to arise in dynamic motions with abrupt directional changes. This phenomenon is perhaps due to dynamic situations demanding high precision from the policy, while these high-precision-requiring states are overshadowed by simpler states in the training data distribution.

Another observation worth noting is the performance variations across the two dataset splits. Since MJPC does not necessitate offline training, its performance remains virtually unchanged between the train and test splits. The slight improvement observed in the test split can be explained by the relative simplicity of motions in the dataset, which largely lacks any dynamic motions like backflips or cartwheels. PHC, on the other hand, demonstrates a much higher error rate on the test set compared to the training set, suggesting its inability to faithfully track motions beyond the training distribution.

Success/Failure Rate. When examining the tracking success, we find that the criterion used by Luo et al. [2021, 2023a], corresponding to $T^{\text{mean}}_{\tau=0.5}$, may be overly lenient. The mean metric, especially when combined with such a high termination threshold, often overlooks failures that would be obvious to human observers. For instance, in tracking pirouettes, simply remaining stationary is sufficient; similarly,

for jumps under 0.5 meters, merely keeping the feet on the ground might be adequate. We believe that adopting $T_\tau^{\max}$ as a termination condition better captures such errors and might be more suitable for assessing the tracking success.

Luo et al. [2021, 2023a] report a scalar success rate, the average of $T_{\tau=0.5}^{\text{mean}}$ over the sequences, for each dataset. To provide a more comprehensive understanding of tracking success, we present success rates for both T_τ^{mean} and $T_\tau^{\max}$, across varying termination thresholds τ. These results are detailed in figures 5.5a and 5.5b. Our primary observation is that MJPC is Pareto dominant over PHC across both datasets. More critically, these results underscore the potential for improvement in these models. It is noteworthy that both methods fail to reach a 95% success rate when evaluated against the max-error termination criterion, $T_{\tau=0.5}^{\max}$. While this criterion is significantly stricter than the mean-error criterion, it is worth considering that having a hand or foot within 0.5 meters of the intended target is, intuitively, not an unreasonable requirement. Achieving higher success rates under this metric is a target we believe is achievable.

5.6 Discussion and Future Work

We demonstrated that MPC outperforms state-of-the-art Reinforcement Learning (RL)-based motion tracking approaches, especially in scenarios involving motions outside the training distribution of the RL models. One of the key advantages of MPC highlighted by our experiments is its flexibility and efficiency. Unlike RL-based methods, which require extensive offline training and hyperparameter tuning, MPC can be deployed quickly and adjusted in real-time, providing a more interactive and iterative development process. This is complemented by the fact that our implementation runs efficiently on standard computing hardware, making advanced motion synthesis and tracking accessible to a broader audience without the need for specialized accelerators like GPUs.

We advocated for better metrics and more thorough evaluations for quantifying the motion tracking performance. The metrics currently used in the field are often

overly forgiving, allowing near-perfect scores without actually resulting in faithful imitation, particularly for dynamic out-of-distribution motions.

The computational demand of MPC, particularly for models with high degrees of freedom, like the `SMPLHumanoid`, can still be significant, limiting its real-time applicability. While our framework achieves near-real-time tracking on simpler models, the computational overhead of vanilla iLQG does not scale particularly well for models with a higher number of degrees of freedom. Incorporating speed enhancements introduced by prior works Han et al. [2016], Russell et al. [2023] into iLQG, and investigating more efficient algorithms and optimizations to reduce the computational load of MPC in general would be a natural future direction.

There is a practical trade-off between MPC and RL depending on both the diversity and number of target motions. If only a small number of motions require tracking or if the motions frequently lie outside the training distribution of an RL policy, it is likely that MPC is a more efficient and straightforward solution to apply. In contrast, for users who need to track thousands of motions that fall well within the training distribution, a learned universal tracking policy can be more cost-effective: although the upfront training cost for RL can be substantial, once a policy is learned, it can generate motions in real time (or faster) for any scenario within its training range, effectively amortizing the initial training expenses over many tasks.

Exploring MPC planners as expert data generators for distilling behaviors into parametric policies within a framework like chapter 4 is an exciting future direction. Distillation through supervised imitation learning could potentially enable a more stable training regime than RL, thus providing real-time performance even on more complex character models without sacrificing motion quality. Our preliminary investigations, not covered in this work, revealed that the temporally-smooth actions — achieved in our case through the regularization of actuator activation change $\dot{\mathbf{a}}$ — are crucial for successful distillation. Without such smoothness, iLQG-generated plans tend to change dramatically between time steps, thereby complicating the learning process for neural networks. While DAgger Ross et al. [2011] enabled us to distill majority of the sequences effectively, we identified challenges with certain

"bottleneck states", some of which appeared, deceptively, in visually extremely simple motions. We hypothesize that these bottleneck states require high precision from the policy, yet their importance is diluted by the predominance of simpler states in the training data distribution.

In conclusion, our work underscores the continued potential of MPC as a powerful tool for physics-based motion tracking, offering significant advantages in terms of flexibility, performance, and accessibility.

6
Future Work and Conclusions

This thesis has presented a variety of algorithms and methods aimed at addressing challenges in the continuous control of embodied agents. Our focus spanned from improving exploration and knowledge transfer for more efficient reinforcement learning to developing a tool for physics-based full-body motion tracking.

We began by addressing the limitations of exploration in model-free RL. Using the *B*ayesian *B*ellman *O*perators (BBO) framework, we derived a practical policy evaluation method and introduced the Bayesian Bellman Actor-Critic (BBAC) algorithm for continuous control. Our experiments demonstrated the algorithm's consistency and convergence properties in the policy evaluation setting. Similarly, in a control setting, BBAC also demonstrated deep and adaptive exploration capabilities driven by the agent's uncertainty about the environment. Our results are encouraging and underscore the potential for wider adoption of model-free Bayesian methods in continuous RL applications. However, the Bayesian methods are not without limitations. While our algorithm performs well in simpler environments, such as the cartpole, our experiments in higher-dimensional domains exposed clear challenges. Evidently, even near-Bayes-optimal exploration strategies will fall short in tabula-rasa scenarios, lacking any prior knowledge.

Departing from the Bayesian perspective, we then explored transfer learning, focusing on the impact of information asymmetry and hierarchy within KL-

regularized transfer in reinforcement learning. Our exploration highlighted the crucial expressivity-transferability trade-off, demonstrating the need for careful selection of information asymmetry levels, often requiring domain knowledge and extensive experimentation. To mitigate the burden of manually tuning the asymmetry, we introduced a method that learns an attention-based mechanism to dynamically adapt the level of information asymmetry. This approach proved effective in a robotic cube manipulation task, adapting dynamically to the requirements in the environment. Despite these advancements, transfer learning poses inherent challenges; the efficacy of skill learning and transfer is heavily influenced by the diversity and distribution of source tasks. Crafting a diverse set of source tasks remains difficult. Consequently, even the best transfer learning methods will encounter limitations in scenarios lacking a sufficiently broad spectrum of experiences from which to generalize.

In chapter 5, we turned our attention towards other kinds of inductive biases suitable for embodied learners. Keyframe motion databases, offering extensive demonstrations particularly from dogs and humans, serve as rich resources from which our artificial embodied agents can draw inspiration. Recent motion tracking literature has predominantly focused on RL methods, which are still challenging to apply in practice. We proposed a practical tool for interactive full-body motion tracking, leveraging traditional model-predictive control methods, and in our experiments we highlighted the shortcomings in contemporary RL-based universal tracking controllers. Our implementation, which operates at interactive rates on commodity laptop hardware, has been made openly available and sets a strong baseline for RL-based tracking controllers on a large-scale motion dataset.

With these contributions, we have sought to deepen the understanding and expand the capabilities of reinforcement learners, particularly for embodied control. Along the way, we introduced both algorithmic and practical advances in exploration for model-free reinforcement learning, skill learning, and physically-simulated character control. Nevertheless, many challenges remain and more work remains to be done.

6.0.1 Future Work

Towards Bayes-optimal exploration. As shown by [Fellows et al., 2023], BBO cannot achieve truly Bayes-optimal exploration behavior but instead solves an approximation of the Bayesian RL objective. Developing methods that more closely approach true Bayesian optimality remains an exciting research direction. Improvements in posterior approximation (especially in higher-dimensional domains) and more sophisticated uncertainty modeling of the Bellman operator could further enhance the efficiency of Bayesian techniques.

Leveraging MPC Experts as Priors. A promising extension of our work would be to incorporate trajectory-optimization experts from chapter 5 as priors for Bayesian agents such as BBO introduced in section 3.3. Likewise, the same experts could serve as rich prior knowledge in a hierarchical skill transfer system like the one presented in section 4.4. As humanoid robots continue to gain technological and commercial interest, exploring such strategies in real-world, dynamic environments could open up significant opportunities for robust control and more diverse tasks.

Expanding Motion Capture Datasets. Moreover, while we focused on curated, high-quality keyframe datasets with no object interaction, future research could exploit broader and noisier sources of motion data, including, for example, video footage from everyday human activities. Automatically extracting usable object-interaction trajectories would allow for training agents on much richer and larger-scale motion datasets. This could prove valuable not only for character animation but also for robotics, where context-rich motion data could guide more adaptive and versatile controllers.

In summary, while the methods presented here advance the methods for efficient exploration, transfer learning, and physics-based character control, these domains are far from being solved. Further progress will require both theoretical and practical contributions to deploy these ideas in increasingly complex and real-world settings.

It is our hope that this work provides a foundation for ongoing efforts, leading to broader adoption and more reliable performance in embodied applications.

Appendices

A

Bayesian Bellman Operators

A.1 Derivations

A.1.1 Posterior Density Derivation

We now derive the posterior density in equation (3.3). We start by deriving the likelihood of the data $\mathcal{D}_\omega^N$ for the i.i.d. case:

$$p_\Phi(\mathcal{D}_\omega^N|\phi) = \prod_{i=1}^{N} \rho(s_i)\pi(a_i|s_i)p(b_i|s_i,a_i,\phi). \tag{A.1}$$

Using our prior $p_\Phi(\phi)$ and Bayes' rule, we can infer the posterior as:

$$\begin{aligned}
p_\Phi(\phi|\mathcal{D}_\omega^N) &= \frac{p(\mathcal{D}_\omega^N|\phi)p_\Phi(\phi)}{\int_\Phi p(\mathcal{D}_\omega^N|\phi)\, dP_\Phi(\phi)} && \text{(A.2)}\\
&= \frac{\prod_{i=1}^N \left(\rho(s_i)\pi(a_i|s_i)p(b_i|s_i,a_i,\phi)\right) p_\Phi(\phi)}{\int_\phi \prod_{i=1}^N \left(\rho(s_i)\pi(a_i|s_i)p(b_i|s_i,a_i,\phi)\right)\, dP_\Phi(\phi)} && \text{(A.3)}\\
&= \frac{\prod_{j=1}^N(\rho(s_j)\pi(a_j|s_j))\prod_{i=1}^N p(b_i|s_i,a_i,\phi)p_\Phi(\phi)}{\int_\phi \prod_{j=1}^N(\rho(s_j)\pi(a_j|s_j))\prod_{i=1}^N p(b_i|s_i,a_i,\phi)\, dP_\Phi(\phi)} && \text{(A.4)}\\
&= \frac{\prod_{j=1}^N(\rho(s_j)\pi(a_j|s_j))\prod_{i=1}^N p(b_i|s_i,a_i,\phi)p_\Phi(\phi)}{\prod_{j=1}^N(\rho(s_j)\pi(a_j|s_j))\int_\phi \prod_{i=1}^N p(b_i|s_i,a_i,\phi)\, dP_\Phi(\phi)} && \text{(A.5)}\\
&= \frac{\prod_{i=1}^N p(b_i|s_i,a_i,\phi)p_\Phi(\phi)}{\int_\phi \prod_{i=1}^N p(b_i|s_i,a_i,\phi)\, dP_\Phi(\phi)}. && \text{(A.6)}
\end{aligned}$$

For sampling from an ergodic Markov chain, we denote the initial state-action density as $p_0(s,a)$ and the transition density as $p(s',a'|s,a)$. For sampling on-policy

these distributions are defined as:

$$p_0(s,a) \coloneqq p_0(s)\pi(a|s) \tag{A.7}$$

$$p(s',a'|s,a) \coloneqq p(s'|s,a)\pi(a'|s'). \tag{A.8}$$

We write our likelihood as:

$$p(\mathcal{D}_\omega^N|\phi) = p_0(s_1,a_1)p(b_1|s_1,a_1,\phi)\prod_{i=2}^{N}\left(p(s_i\,a_i|s_{i-1},a_{i-1})p(b_i|s_i,a_i,\phi)\right) \tag{A.9}$$

$$= p_0(s_1,a_1)\prod_{j=2}^{N}p(s_j\,a_j|s_{j-1},a_{j-1})\prod_{i=1}^{N}p(b_i|s_i,a_i,\phi) \tag{A.10}$$

$$= p(S_N,A_N)\prod_{i=1}^{N}p(b_i|s_i,a_i,\phi), \tag{A.11}$$

where $S_N \coloneqq \{s_1,...s_N\}$, $A_N \coloneqq \{a_1,...a_N\}$ and

$$p(S_N,A_N) = p_0(s_1,a_1)\prod_{j=2}^{N}p(s_j\,a_j|s_{j-1},a_{j-1}). \tag{A.12}$$

We now infer the posterior using Bayes' rule:

$$p_\Phi(\phi|\mathcal{D}_\omega^N) = \frac{p(\mathcal{D}_\omega^N|\phi)p_\Phi(\phi)}{\int_\Phi p(\mathcal{D}_\omega^N|\phi)\,dP_\Phi(\phi)} \tag{A.13}$$

$$= \frac{p(S_N,A_N)p(b_i|s_i,a_i,\phi)p_\Phi(\phi)}{p(S_N,A_N)\int_\phi\prod_{i=1}^{N}p(b_i|s_i,a_i,\phi)\,dP_\Phi(\phi)} \tag{A.14}$$

$$= \frac{\prod_{i=1}^{N}p(b_i|s_i,a_i,\phi)p_\Phi(\phi)}{\int_\phi\prod_{i=1}^{N}p(b_i|s_i,a_i,\phi)\,dP_\Phi(\phi)}, \tag{A.15}$$

which has the same form as the i.i.d. case in equation (A.6).

A.1.2 MSBBE Gradient Derivation

We now derive an analytic form for the derivative of the MSBBE in equation (3.6). Starting from the definition of the MSBBE:

$$\nabla_\omega\text{MSBBE}_N(\omega) = \nabla_\omega\left\|\hat{Q}_\omega - \mathcal{B}_{\omega,N}\right\|_{\rho,\pi}^2 \tag{A.16}$$

$$= \frac{1}{2}\nabla_\omega\mathbb{E}_{\rho,\pi}\left[(\hat{Q}_\omega - \mathcal{B}_{\omega,N})^2\right], \tag{A.17}$$

$$= \mathbb{E}_{\rho,\pi}\left[(\hat{Q}_\omega - \mathcal{B}_{\omega,N})(\nabla_\omega\hat{Q}_\omega - \nabla_\omega\mathcal{B}_{\omega,N})\right], \tag{A.18}$$

Substituting for the definition of the Bayesian Bellman operator from equation (3.4) yields our desired result:

$$\nabla_\omega \mathrm{MSBBE}_N(\omega) \tag{A.19}$$

$$= \mathbb{E}_{\rho,\pi}\left[\left(\hat{Q}_\omega - \mathbb{E}_{P_\Phi(\phi|\mathcal{D}_\omega^N)}\left[\hat{B}_\phi\right]\right)\left(\nabla_\omega \hat{Q}_\omega - \nabla_\omega \mathbb{E}_{P_\Phi(\phi|\mathcal{D}_\omega^N)}\left[\hat{B}_\phi\right]\right)\right] \tag{A.20}$$

$$= \mathbb{E}_{\rho,\pi}\left[\left(\hat{Q}_\omega - \mathbb{E}_{P_\Phi(\phi|\mathcal{D}_\omega^N)}\left[\hat{B}_\phi\right]\right)\left(\nabla_\omega \hat{Q}_\omega - \mathbb{E}_{P_\Phi(\phi|\mathcal{D}_\omega^N)}\left[\hat{B}_\phi \nabla_\omega \log p_\Phi(\phi|\mathcal{D}_\omega^N)\right]\right)\right]. \tag{A.21}$$

A.1.3 Gaussian BBO Derivation

Now, using a Gaussian model:

$$P(b|s,a,\phi) \coloneqq \mathcal{N}(\hat{B}_\phi(s,a), \sigma^2), \tag{A.22}$$

and defining the log-normalization constant as $c_{\text{norm}} \coloneqq \log \int_\phi \prod_{i=1}^N p(b_i|s_i,a_i,\phi)\, dP_\Phi(\phi)$, we can derive the exact form of the log-posterior given in equation (3.7):

$$p(\phi|\mathcal{D}_\omega^N) = \exp(-c_{\text{norm}}) \prod_{i=1}^N p(b_i|s_i,a_i,\phi) p(\phi) \tag{A.23}$$

$$= \exp(-c_{\text{norm}}) \prod_{i=1}^N \exp\left(-\frac{1}{2\sigma^2}(b_i - \hat{B}_\phi(s_i,a_i))^2\right) \exp\left(-R(\phi)\right) \tag{A.24}$$

$$= \exp\left(-c_{\text{norm}} - \frac{1}{2\sigma^2}\sum_{i=1}^N (b_i - \hat{B}_\phi(s_i,a_i))^2 - R(\phi)\right) \tag{A.25}$$

$$\implies -\log p(\phi|\mathcal{D}_\omega^N) = c_{\text{norm}} + \sum_{i=1}^N \frac{(b_i - \hat{B}_\phi(s_i,a_i))^2}{2\sigma^2} + R(\phi), \tag{A.26}$$

as required.

A.2 BBO Policy Evaluation Experiment Details

A.2.1 Tsitsiklis' Triangle Counterexample

The three-state Tsitsiklis' Triangle MDP [Tsitsiklis and Van Roy, 1997] is designed to prove divergence of `TD` methods with nonlinear function approximators. The purpose of this experiment is to empirically validate the convergence properties of nonlinear BBO algorithms.

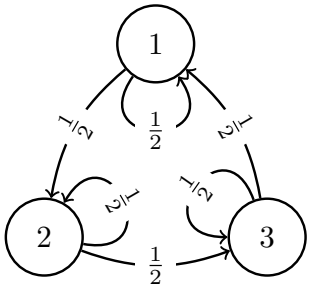


Figure A.1: Tsitsiklis' Triangle MDP.

Environment and Value Function

The environment, illustrated in figure A.1, consists of the state space $S = \{1, 2, 3\}$ and the action-independent transition kernel

$$P \coloneqq [p(s' = j|\cdot, s = i)]_{i,j} = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}. \tag{A.27}$$

Let $\hat{V}(\omega) \coloneqq [\hat{V}_\omega(s = 1), \hat{V}_\omega(s = 2), \hat{V}_\omega(s = 3)]^\top$ be the value function vector. It can be shown that any $\hat{V}(\omega)$ parameterized by $\omega \in \mathbb{R}$ and satisfying the linear dynamical system

$$\frac{d\hat{V}(\omega)}{d\omega} = (Q + \epsilon I)\,\hat{V}(\omega), \tag{A.28}$$

with the condition that $\hat{V}(0)^\top 1 = 0$ where $\epsilon > 0$ is a small constant and:

$$Q \coloneqq \begin{bmatrix} 1 & \frac{1}{2} & \frac{3}{2} \\ \frac{3}{2} & 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{3}{2} & 1 \end{bmatrix}, \tag{A.29}$$

diverges when updated using the `TD` algorithm [Tsitsiklis and Van Roy, 1997]. To be consistent with the `TDC` and `GTD2` algorithms by Bhatnagar et al. [2009] we choose:

$$\begin{aligned} \hat{V}(\omega) &= \exp(\epsilon\omega)(a\cos(\lambda\omega) - b\sin(\lambda\omega)) \\ \text{with } a &= (-14.9996, -35.0002, 50.0004), \\ b &= (-49.0753, 37.5278, 11.5469), \\ \lambda &= \sqrt{3}/2, \text{ and} \\ \epsilon &= 10^{-2}, \end{aligned}$$

as a solution to equation (A.28) (using the values that we received from the authors' implementation). We set the discount $\gamma = 0.9$ and normalize the gradient steps to stabilize the updates. Each update batch includes all 6 environment transitions.

Hyperparameters

We search over the learning rates for all the algorithms. For `TDC`, and `GTD2`, we begin with a coarse grid search with values of $\{1e{-}3, 1e{-}2, 1e{-}1, 1e0\}$, followed by finer search of $\{1e{-}1, 2e{-}1, ..., 9e{-}1, 1e0\}$. For `TD(0)`, we manually search around the learning rate used in Bhatnagar et al. [2009]. We do not search over BBO's prior loss weight and set it to 1.0. The final hyperparameters are presented in table A.1.

Table A.1: Hyperparameters for reported Tsitsiklis Triangle experiments.

Method	Parameter	Value
BBO	Lower-level learning rate	8e-1
	Upper-level learning rate	1e-1
`TD(0)`	Learning rate	2e-3
`TDC`	Fast timescale learning rate	1e0
	Slow timescale learning rate	1e-1
`GTD2`	Fast timescale learning rate	8e-1
	Slow timescale learning rate	1e-1

Results

As can be seen in figure 3.3, BBO converges to the optimal solution similarly to prior convergent nonlinear methods, `TDC` and `GTD2` Bhatnagar et al. [2009], while `TD(0)`, as expected, diverges. These results verify the convergence properties of the proposed nonlinear BBO algorithms.

A.2.2 Neural Network Function Approximators

Environments

We consider three environments commonly used in policy evaluation literature: `100-Link Pendulum` [Dann et al., 2014] with 200-dimensional continuous observation space, `Puddle World` [Boyan and Moore, 1995] with 2-dimensional continuous observation space, and the continuous variant of `Mountain Car` [Boyan and Moore, 1995][1] with 2-dimensional continuous observation space.

[1]We use the `MountainCar-Continuous-v0` implementation from the OpenAI Gym suite [Brockman et al., 2016]

Datasets

The datasets used for evaluating the policies consist of 2e4, 2e4, and 5e4 on-policy transitions for `Puddle World`, `Mountain Car`, and `100-Link Pendulum`, respectively. For `Puddle World` and `Mountain Car`, each transition is sampled independently by resetting the state uniformly at random in the state space after each transition. `Puddle World` and `Mountain Car` observations are normalized to range $[-1, +1]$.

Policies. For `Puddle World` and `Mountain Car` experiments, we run the policy evaluation using a simple, non-optimal policy. In `Puddle World`, the policy selects either up or down action uniformly at random. In `Mountain Car`, the right action is chosen when velocity > 0, and otherwise the left action. The `100-Link Pendulum` experiments, on the other hand, use an optimal policy obtained with dynamic programming, similar to the policy used in the corresponding linear experiments. See Dann et al. [2014] for details.

Ground-truth Value Functions. For `Puddle World` and `Mountain Car`, the ground-truth value function, for which the mean squared errors are computed, is obtained for a set of 625 evenly-spaced states (25×25 grid) in the 2D state space. We reset the agent to each of the 625 states 1000 times, rolling it out and computing the cumulative sum of rewards for up to 1000 steps, finally averaging over the 1000 resets. The `100-Link Pendulum` environment is a linear-quadratic MDP, for which we obtain the exact values for 5000 states, as is done by Dann et al. [2014]. All the value functions are computed with the same discount factor that is used for training the approximate value functions.

Network Structure

All the experiments use a single-layer feedforward network with hidden layer size of 256 for `Puddle World` and `Mountain Car` and 512 for `100-Link Pendulum`, unless otherwise stated. `TDC` uses a *tanh* activation whereas other algorithms use *relu*,.

Training Details

The discount factor is set to 0.98 for all tasks. The optimization is carried out with Adam optimizer [Kingma and Ba, 2015] using randomly sampled mini-batches of size 512. The hyperparameter searches are done using 3 datasets and the final results are averaged over 24 separate datasets and seeds. The runs take about 15 minutes for BBO variants, 7 minutes for `TD(0)`, and 85 minutes for `TDC` per 100k steps on a standard desktop machine.

Hyperparameters

We use the same grid search for the hyperparameters across all environments. The hyperparameters with their evaluated values for each algorithm are presented in table A.2 for BBO, table A.3 for `TD(0)`, and table A.4 for `TDC`. For BBO, instead of searching over the full grid at once, we first perform a coarse grid search for learning rates without using a prior, then perform another denser search around the best values (with values still given in the table A.2), and finally perform a grid search over the lower-level gradient steps per training step and prior values (when applicable). For `TDC` and `TD(0)`, we perform a full grid search over the listed values.

Table A.2: BBO hyperparameter grid

Hyperparameter	Grid values
Learning rates	$\{1e{-}6, 3e{-}6, 1e{-}5, \ldots, 1e{-}1, 3e{-}1, 1e0\}$
Weight for the prior loss $(1/\sigma_0^2)^*$	$\{1e{-}4, 3e{-}4, 1e{-}3, \ldots, 1e{-}1, 3e{-}1, 1e0\}$
Lower-level steps / training step	$\{1, 5, 10, 20\}$

* For `100-Link Pendulum`, we also include $\{2.5e{-}1, 5e{-}1, 7.5e{-}1\}$.

Table A.3: `TD(0)` hyperparameter grid

Hyperparameter	Grid values
Learning rate*	$\{1e{-}6, 3e{-}6, 1e{-}5, \ldots, 1e{-}1, 3e{-}1, 1e0\}$

* For `100-Link Pendulum`, we also include $\{1e{-}7, 3e{-}7\}$.

The final hyperparameters for each method and task are presented in table A.5. The values are chosen by manually picking the best-performing set from the

Table A.4: `TDC` hyperparameter grid

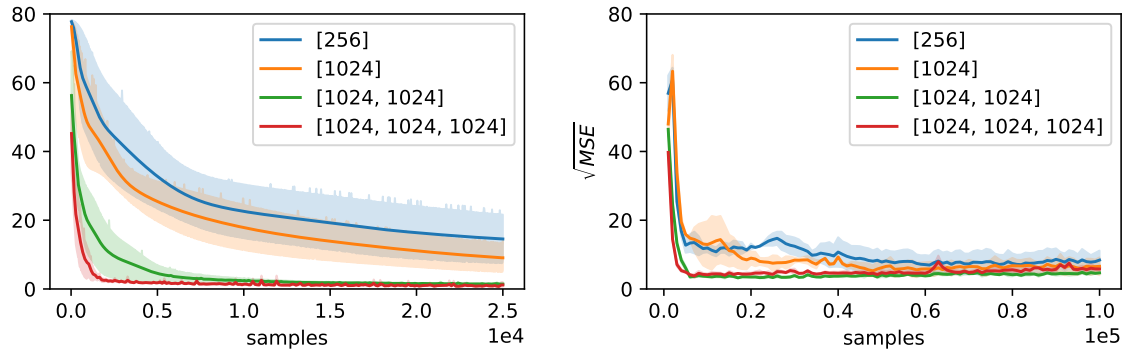
Hyperparameter	Grid values
Fast timescale learning rate	$\{1e{-}6, 3e{-}6, 1e{-}5, \ldots, 1e{-}1, 3e{-}1, 1e0\}$
Slow timescale learning rate	$\{1e{-}6, 3e{-}6, 1e{-}5, \ldots, 1e{-}1, 3e{-}1, 1e0\}$

Table A.5: Hyperparameters for reported nonlinear policy evaluation results.

Method	Parameter	Environment		
		`100-Link Pendulum`	`Puddle World`	`Mountain Car`
Gradient BBO w/ prior	Upper-level learning rate	1e-3	3e-2	1e-2
	Lower-level learning rate	1e-4	1e-2	3e-3
	Prior loss weight	5e-1	3e-4	1e-1
	Lower-level steps / training step	20	10	10
Gradient BBO w/o prior	Upper-level learning rate	1e-4	1e-3	1e-2
	Lower-level learning rate	3e-4	1e-2	3e-3
	Lower-level steps / training step	20	10	10
Direct BBO w/ prior	Learning rate	3e-7	1e-3	3e-4
	Prior loss weight	1.0	3e-4	1e-1
`TD(0)`	Learning rate	3e-7	3e-3	3e-4
`TDC`	Fast timescale learning rate	1e-3	1e-3	1e-2
	Slow timescale learning rate	1e-4	1e-3	1e-5

hyperparameter search based on the final MSE, and the final results are reported for 24 holdout datasets.

A.2.3 Residual MSE in Puddle World Experiments



(a) Supervised Oracle **(b)** BBO

Figure A.2: Learning curves for supervised oracle (left, a) and BBO (right, b) with different hidden layer sizes in policy evaluation task. The oracle is trained with supervised learning to minimize the MSE directly, and it achieves near-zero error with higher-capacity models. For BBO, incresing the network capacity does not eliminate the residual error. Note the different scale of the horizontal axis.

In section 3.6.1, we presented results in three non-linear policy evaluation tasks. A curious reader might wonder where the residual error in the `Puddle World` result comes from.

As previously described, the results in figure 3.4 present the root of the mean-squared error, $\text{MSE}(\omega) \coloneqq \|V^\pi - \hat{V}_\omega\|_2^2$, between the true value function V^π and the learned value function $\hat{V}_\omega$. A standard assumption in policy evaluation tasks is that we cannot evaluate the MSE value during training time. If that were possible, we could just resort to minimizing the MSE and forget the temporal-difference updates altogether. In practice, however, we have access to only a dataset of state-transition samples of the environment, and the objectives based on temporal-difference updates act only as a practical proxy for the true MSE error.

Although in principle minimizing MSBBE until zero would imply perfect consistency with the true value function and thus zero MSE, it is well known that in practice TD fixed points — that is, the stable solution that TD learning approaches converge to — may not lead to a non-zero MSE at convergence. We briefly recap some of these practical factors next.

Parametric function approximators. With parametric function approximators, like neural networks in our case, a simple culprit can be the limitation of the function class. In `Puddle World`, for instance, the value function has relatively sharp transitions (especially at the edges of "puddles") that can be difficult to capture precisely without extensive network capacity. This limitation can lead to irreducible MSE.

We ran two experiments to investigate whether the model capacity is indeed the limiting factor. First, we trained a neural network value functions of varying size in a supervised learning setting, where we minimize the MSE loss directly, using the privileged "oracle" value function during training (which for the actual policy evaluation task is only used during evaluation.) Second, we do a similar ablation for BBO, varying the network capacity to understand its effect on the MSE.

The results are presented in figure A.2. Left (figure A.2a), the oracle model clearly benefits from higher model capacity: the smaller, shallower networks ([256] and [1024]) suffer from both higher variance across the seeds and also do not converge within the training window. With deeper networks ($[1024, 1024]$ and $[1024, 1024, 1024]$), the MSE is quickly driven down to near-zero values (with final RMSE $<< 0.5$.). Similar analysis for BBO (figure A.2b) shows that the network capacity alone is unable to drive the residual RMSE to zero: while the RMSE improves when increasing the network size from [256] to [1024], and achieving a slightly lower minimum value with $[1024, 1024]$ network, increasing the network size further to $[1024, 1024, 1024]$ increases the RMSE. This suggests that the residual RMSE cannot be explained with the network capacity alone.

Regularization and prior influence. BBO, as is typical with Bayesian approaches, employs a prior that biases the solution space. This regularization helps the algorithm avoid overfitting but can also nudge the solution away from the exact empirical optimum, thus leaving a nonzero gap in MSE. In other words, the "best compromise" between data fitting and prior regularization might never

be exactly zero error. We investigated removing the prior term but observed no meaningful benefit in the RMSE at convergence.

Finite sampling and gradient-based optimization used by BBO can also lead to small residual differences in the learned solution. Hence, the convergence to a local minimizer of MSBBE while still having slight MSE above zero.

Stochasticity in updates or environment. The `Puddle World` task [Boyan and Moore, 1995] is inherently stochastic, the sampling of transitions can introduce noise, particularly with the finite, fixed dataset as used in our experiments.

A.3 BBO Continuous Control Experiment Details

This section provides further details and analysis of the BBAC algorithm and control experiments presented in section 3.6. We refer to $\lambda \coloneqq \frac{1}{\sigma_0^2}$ as the regularization weight and σ^2 as the prior scale. We first investigate BBAC's behavior in the `MountainCar-Continuous-v0` task, where the environment's simplicity and low-dimensionality allows us to visually analyze the behavior of the randomized prior ensemble. We then analyze BBAC's sensitivity to randomized prior hyperparameters in a slightly modified version of DeepMind Control Suite's [Tassa et al., 2018] `cartpole-swingup_sparse` environment. Finally, we highlight the limitations of BBAC in higher-dimensional `Humanoid-v3` environment from [Brockman et al., 2016].

To improve the exploration capability of SAC in the challenging domains, we also introduce a variant of SAC called SAC* which uses a single Q-function to avoid *pessimistic underexploration* [Ciosek et al., 2019]. Here, instead of learning two soft Q-function approximators independently and choosing the minimum for the actor and critic gradient updates as specified by SAC, we train a single soft Q-function and use updates (6) and (13) of Haarnoja et al. [2018b] directly. This also reduces the inductive bias that SAC has for solving tasks with dense reward structures, making the comparison against BBAC fairer.

All the experiments in this section use the same hyperparameters from table A.6. The policies and function approximators are parameterized as fully-connected neural networks.

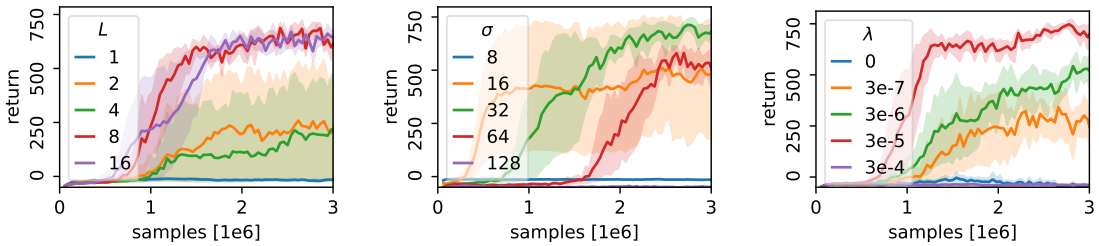
Table A.6: Common Hyperparameters for BBAC and SAC.

Hyperparameter	Value
Optimizer	Adam
Learning rates	3e−4
Discount	0.99
Replay buffer size	1e6
Nonlinearity	ReLU
Hidden layers	2
Hidden units/layer	256
Batch size	256
Target smoothing coefficient	5e−3
Training steps per environment step	1

A.3.1 BBAC MountainCar-Continuous-v0

In order to qualitatively measure how the agents explore the state space, we analyze the evolution of critics and coverage of state space throughout learning. We use the standard `MountainCar-Continuous-v0` implementation from the OpenAI Gym suite [Brockman et al., 2016], and run each of the algorithms for $1e5$ timesteps, while checkpointing the replay pool and the critics on pre-specified intervals.

For the final SAC runs, we set the target entropy using the heuristic provided in [Haarnoja et al., 2018d] which is the negative number of action dimensions, i.e. -1 in this case. We also tested target entropy values $\{-8, -4, -2, 0, \frac{1}{2}, 1\}$, all of which behaved similarly as the default target. From the 35 runs with these 7 different entropies, a total of 3 seeds were able to achieve the goal. As our experiments are carried out in a tabula rasa setting where the prior functions are drawn as randomly initialized neural networks, we find that the choice of prior parameters affects the speed with which BBAC solves the tasks. BBAC is relatively insensitive to the randomized prior hyperparameters in this environment, as long as the ensemble size $L \geq 4$, and for the final results we use ensemble size $L = 8$, prior scale $\sigma = 100$, and regularization weight $\lambda = 3\text{e}{-5}$.



(a) Ensemble size L. λ=3e−5, σ=32.

(b) Prior scale σ. L=8, λ=3e−5.

(c) Regularization weight λ. L=8, σ=32.

Figure A.3: Evaluation of BBAC's sensitivity to randomized prior hyperparameters. We vary (a) ensemble size L, (b) prior scale σ, and (c) regularization weight λ, while keeping other hyperparameters fixed.

A.3.2 BBAC Cartpole

We further investigate BBAC's sensitivity to the choice of randomized prior hyperparameters — ensemble size (L), prior scale (σ), and regularization weight (λ) — in a slightly modified version of DeepMind Control Suite's [Tassa et al., 2018] `cartpole-swingup_sparse` domain, as a continuous analog to the one presented in [Osband et al., 2019b].

The original `cartpole-swingup_sparse` is modified such that the reward function includes a control cost of $0.1a_t$ for action a_t on each timestep t, and the agent receives a positive reward of 1 only when both the cart is controllably centered and the pole is controllably upright. That is, the reward function $r(s_t, a_t)$ for action a_t at state $s_t = (\cos(\theta_t), \sin(\theta_t), \dot{\theta}_t, x_t, \dot{x}_t)$ is given by:

$$r(s_t, a_t) = -0.1|a_t| + (\mathbb{1}(|x_t| < 0.1) \cdot \mathbb{1}(0.95 < \cos(\theta_t)) \cdot \mathbb{1}(|\dot{x}_t| < 1) \cdot \mathbb{1}(|\dot{\theta}_t| < 1)). \tag{A.30}$$

The pole is initialized to a stationary downright position, and the motor is not strong enough to turn the pole upright on a single pass, meaning that in order to reach the goal, the agent has to build momentum by moving the cart and swinging the pole back and forth. This requires executing costly actions for more than a hundred steps, making the exploration problem non-trivial.

SAC's performance, shown in figure 3.5b, confirms that naïve exploration strategies, such as noisy actions incorporated by maximum-entropy reinforcement

learning, eschew costly exploration actions needed for task completion and converge to a sub-optimal strategy. BBAC on the other hand is able to consistently solve the task.

The results for BBAC's hyperparameter sensitivity are presented in figure A.3. Increasing the ensemble size (L; figure A.3a) improves the likelihood of solving the task and, as expected, this effect plateaus and the task can be consistently solved with $L \geq 8$. In the case of both prior scale (σ; figure A.3b) and regularization weight (λ; figure A.3c), too small values limit the exploration and there is a sweet spot where task performance and exploration are well-balanced. While higher values of prior scale are unable to solve the task within the $3e6$ time steps shown here, we expect them to eventually converge to the optimal solution, whereas higher values of regularization weight are likely to constrain the learning too much to allow convergence.

All our experiments are carried out in a tabula rasa setting where the prior functions are drawn as randomly-initialized neural networks, which is why the effect of the hyperparameter choice to the speed of learning is expected. The range of working hyperparameters is relatively wide and easy to tune, however, and in a real-world scenario, we might have access to prior knowledge on the task, for example through transfer learning, which would further simplify the choice of BBAC's hyperparameters.

A.3.3 BBAC Humanoid

For *dense*-reward tasks, we use the standard `Humanoid-v3` implementation from the OpenAI Gym suite [Brockman et al., 2016], which provides a dense set of shaping terms. In the *sparse*-reward version, we remove all the positive shaping terms and disable premature episode termination: the agent receives a positive reward only if it has moved at least five meters from the origin.

We train each trial for $1e7$ timesteps. We ran a sweep for the prior scale values $\sigma \in \{8, 16, 32, 64\}$ and regularization weight values $\lambda \in \{3\text{e}{-6}, 3\text{e}{-5}, 3\text{e}{-4}\}$. The final results for RP-BBAC use ensemble size $L = 8$, prior scale $\sigma = 32$, and

regularization weight $\lambda = 3\text{e}{-5}$. The hyperparameters and results for SAC are identical to those from the original paper [Haarnoja et al., 2018b]. The reported results include five trials with independent seeds for both algorithms. Not a single trial in our sparse experiments managed to achieve the sparse reward.

B

Priors, Hierarchy, and Information Asymmetry for Skill Transfer in Reinforcement Learning

B.1 Covariate Shift and Knowledge Distillation

In this section, we provide proofs for the theory introduced in chapter 4.

B.1.1 Proof of Theorem 4.3.1 (Covariate Shift)

Theorem 4.3.1 (Covariate Shift)**.** *The more random variables a network depends on, the larger the covariate shift (here represented by KL-divergence) encountered across sequential tasks. That is, for two distributions p and q, and inputs $\mathbf{b}$ and $\mathbf{c}$ with $\mathbf{c} \subset \mathbf{b}$:*

$$\mathrm{D}_{\mathrm{KL}}(p(\mathbf{b}) \parallel q(\mathbf{b})) \geq \mathrm{D}_{\mathrm{KL}}(p(\mathbf{c}) \parallel q(\mathbf{c})).$$

Proof of theorem 4.3.1.

$$\begin{aligned}
\mathrm{D}_{\mathrm{KL}}(p(\mathbf{b}) \parallel q(\mathbf{b})) &= \mathbb{E}_{p(\mathbf{b})}\left[\log\left(\frac{p(\mathbf{b})}{q(\mathbf{b})}\right)\right] \\
&= \mathbb{E}_{p(\mathbf{d}|\mathbf{c})\cdot p(\mathbf{c})}\left[\log\left(\frac{p(\mathbf{d}|\mathbf{c})\cdot p(\mathbf{c})}{q(\mathbf{d}|\mathbf{c})\cdot q(\mathbf{c})}\right)\right] \quad \textit{with} \quad \mathbf{d} \in \mathbf{b} \setminus \mathbf{c} \\
&= \mathbb{E}_{p(\mathbf{c})}\left[\mathbb{E}_{p(\mathbf{d}|\mathbf{c})}\left[1\right] \cdot \log\left(\frac{p(\mathbf{c})}{q(\mathbf{c})}\right)\right] + \mathbb{E}_{p(\mathbf{c})}\left[\mathbb{E}_{p(\mathbf{d}|\mathbf{c})}\left[\log\left(\frac{p(\mathbf{d}|\mathbf{c})}{q(\mathbf{d}|\mathbf{c})}\right)\right]\right] \\
&= \mathrm{D}_{\mathrm{KL}}(p(\mathbf{c}) \parallel q(\mathbf{c})) + \mathbb{E}_{p(\mathbf{c})}\left[\mathrm{D}_{\mathrm{KL}}(p(\mathbf{d}|\mathbf{c}) \parallel q(\mathbf{d}|\mathbf{c}))\right] \\
&\geq \mathrm{D}_{\mathrm{KL}}(p(\mathbf{c}) \parallel q(\mathbf{c})) \quad \textit{given} \quad \mathbb{E}_{p(\mathbf{c})}\left[\mathrm{D}_{\mathrm{KL}}(p(\mathbf{d}|\mathbf{c}) \parallel q(\mathbf{d}|\mathbf{c}))\right] \geq 0
\end{aligned} \tag{B.1}$$

□

B.1.2 Proof of Theorem 4.3.2 (Knowledge Distillation)

Theorem 4.3.2 (Knowledge Distillation)**.** *The more random variables a network depends on, the greater its ability to distill knowledge in the expectation (reflected by the expected KL-divergence between the network's output distribution and a target distribution). Specifically, for a target distribution p and a network q with outputs $\mathbf{a}$ and possible inputs $\mathbf{b}$, $\mathbf{c}$, $\mathbf{d}$, such that $\mathbf{b} = (b_0, b_1, ..., b_n)$, $\mathbf{d} \subset \mathbf{c} \subset \mathbf{b}$, $\mathbf{e} \in \mathbf{d} \setminus \mathbf{c}$:*

$$\mathbb{E}_{q(\mathbf{e}|\mathbf{d})}\left[\mathrm{D}_{\mathrm{KL}}(p(\mathbf{a}|\mathbf{b}) \parallel q(\mathbf{a}|\mathbf{c}))\right] \leq \mathrm{D}_{\mathrm{KL}}(p(\mathbf{a}|\mathbf{b}) \parallel q(\mathbf{a}|\mathbf{d})).$$

Proof of theorem 4.3.2.

$$\begin{aligned}
\mathrm{D}_{\mathrm{KL}}(p(\mathbf{a}|\mathbf{b}) \parallel q(\mathbf{a}|\mathbf{d})) &= \mathbb{E}_{p(\mathbf{a}|\mathbf{b})}\left[\log\left(\frac{p(\mathbf{a}|\mathbf{b})}{q(\mathbf{a}|\mathbf{d})}\right)\right] \\
&= \mathbb{E}_{p(\mathbf{a}|\mathbf{b})}\left[\log p(\mathbf{a}|\mathbf{b}) - \log \mathbb{E}_{q(\mathbf{e}|\mathbf{d})}\left[q(\mathbf{a}|\mathbf{c})\right]\right] \quad \textit{with} \quad \mathbf{e} \in \mathbf{d} \setminus \mathbf{c} \\
&\geq \mathbb{E}_{p(\mathbf{a}|\mathbf{b})\cdot q(\mathbf{e}|\mathbf{d})}\left[\log\left(\frac{p(\mathbf{a}|\mathbf{b})}{q(\mathbf{a}|\mathbf{c})}\right)\right] \quad \textit{given Jensen's Inequality} \\
&= \mathbb{E}_{q(\mathbf{e}|\mathbf{d})}\left[\mathrm{D}_{\mathrm{KL}}(p(\mathbf{a}|\mathbf{b}) \parallel q(\mathbf{a}|\mathbf{c}))\right]
\end{aligned} \tag{B.2}$$

□

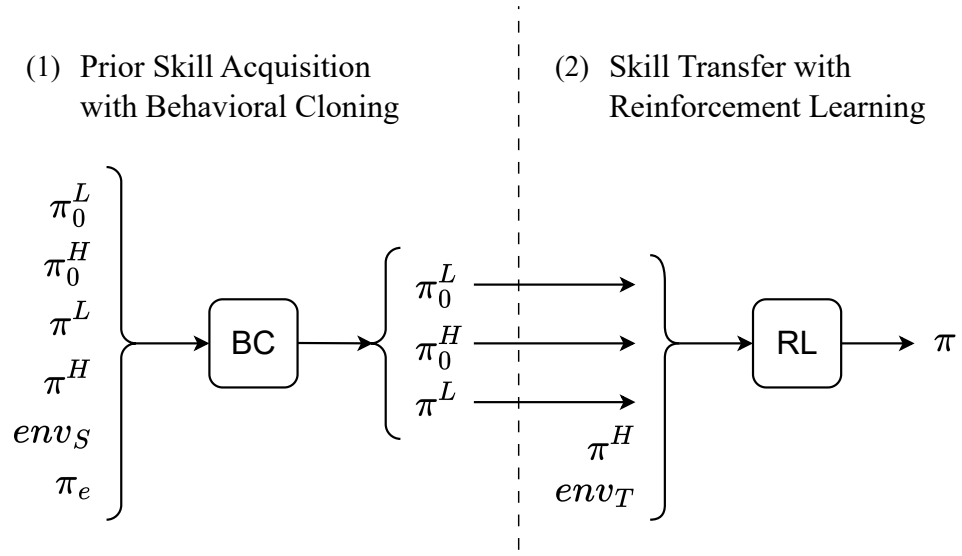


Figure B.1: Schematic of the APES experimental regime.

B.2 Method Details

B.2.1 Training Regime

This section provides details on our method and the training regime used throughout the main sections. We begin by algorithmically describing the training setup and then follow with a discussion on the variational behavioral cloning and reinforcement learning setups in appendix B.2.2, as well as on the critic learning in appendix B.2.3.

Algorithm 4 and figure B.1 provide pseudo-code and a visual schematic of the APES training regime. As described in the main text, the training regime is split into two phases: the prior skill acquisition phase using behavioral cloning and transfer learning with reinforcement learning. The behavioral cloning phase adopts DAgger [Ross et al., 2011] for improved learning efficiency and stability, and produces a prior policy $\pi_0 = \{\pi_0^L, \pi_0^H\}$ as a result of expert imitation in the source environment env_S. The prior is then transferred over to the target environment env_T and we learn a new policy π^H.

Both the behavioral cloning and reinforcement learning phases use the experience collection routine described in algorithm 5, which implements the DAgger logic of randomly sampling the environment with either the expert or the learned policy.

Algorithm 4 APES Training Regime

Input: Policy π, priors $\pi_0 = \{\pi_i\}_{i\in\{0,\dots,N\}}$, critics $Q_{k\in\{1,2\}}$, replay buffers R_{bc} for behavioral cloning and R_{rl} for reinforcement learning, DAGGER rate r, source and transfer environments env_S and env_T.
Output: Trained policy π^H.

▷ Train prior with Behavioral Cloning (BC).
Initialize π, π_0, R_{bc}, r, env_S
for each BC training step **do**
    R_{bc}, $\text{env}_\text{S} \leftarrow$ COLLECT(π, R_{bc}, env_S, True, r) ▷ Algorithm 5
    π, $\pi_0 \leftarrow$ BC_UPDATE(π, π_0, R_{bc}) ▷ Stop $\pi_0 \to \pi$ gradient. Equation (4.6)
end for

▷ Transfer and train π^H with Reinforcement Learning (RL).
Reinitialize π^H. Initialize $Q_{k\in\{1,2\}}$, R_{rl}, env_T.
for each RL training step **do**
    R_{rl}, $\text{env}_\text{T} \leftarrow$ COLLECT(π, R_{rl}, env_T, False, 1) ▷ Algorithm 5
    $\pi^H \leftarrow$ RL_POLICY_UPDATE(π, π_0, R_{rl}) ▷ Equation (4.4)
    $Q_k \leftarrow$ RL_CRITIC_UPDATE(Q_k, π, R_{rl}) ▷ Equation (B.9)
end for

Appendix B.3 provides further details on the hyperparameters and model architectures used in our experiments.

B.2.2 Variational Behavioral Cloning and Reinforcement Learning

For simplicity and broader applicability, this subsection momentarily omits the information gating function objective, $IGF(\mathbf{x}_k)$, specific to APES. However, integrating these derivations with the APES framework remains straightforward.

Behavioral Cloning (BC) and KL-Regularized Reinforcement Learning (RL) share many similarities when viewed through the lens of Variational Inference, especially within hierarchical model structures. The behavioral cloning objective is defined as:

$$\mathcal{J}_{\text{BC}}(\pi, \{\pi_i\}^{i\in I_{\text{BC}}}) = -\sum_{i\in I_{\text{BC}}} \text{D}_{\text{KL}}(\pi(\tau) \parallel \pi_i(\tau)) \tag{B.3}$$

Here, $\mathcal{J}_{\text{BC}}$ corresponds to the KL-divergence between trajectories from policy π and priors π_i, with $i \in I_{\text{BC}} = \{0, u, e\}$ representing the set of learned, uniform, and

Algorithm 5 Collect Experience

Input: Policy π_i, replay Buffer R_j, environment *env*, DAgger Flag DAgger (default: False), DAgger Rate r (default: 1).
Output: Updated replay buffer R_j, updated environment *env*.

▷ Collect experience from π_i or π_e, apply DAgger if instructed, and update R_j, *env*.
function Collect(π_i, R_j, *env*, DAgger, r)
    $\mathbf{x} \leftarrow env$.Observation()
    $\pi_e \leftarrow env$.Expert()
    $\mathbf{a}_i \leftarrow \pi_i(\mathbf{x})$
    $\mathbf{a}_e \leftarrow \pi_e(\mathbf{x})$
    $\mathbf{a} \leftarrow$ Bernoulli($[r, 1-r]$, $[\mathbf{a}_i, \mathbf{a}_e]$)
    $\mathbf{x}', r_k, env \leftarrow env$.Step($\mathbf{a}$)
    if DAgger **then**
        $\mathbf{a}_f \leftarrow \mathbf{a}_e$
    else
        $\mathbf{a}_f \leftarrow \mathbf{a}_i$
    end if
    $R_j \leftarrow R_j$.Update($\mathbf{x}, \mathbf{a}_f, r_k, \mathbf{x}'$)
    return R_j, *env*
end function

expert priors. Contrary to what is usually evaluated in the literature [Pertsch et al., 2020], considering only the expert prior in our formulation yields the reverse KL-divergence.

The reinforcement learning objective is defined as:

$$\mathcal{J}_{\text{RL}}\left(\pi, \{\pi_i\}^{i \in I_{\text{RL}}}\right) = \mathbb{E}_{\pi,\tau}\left[R(\tau)\right] - \sum_{i \in I_{\text{RL}}} \text{D}_{\text{KL}}(\pi(\tau) \parallel \pi_i(\tau)) \tag{B.4}$$

which corresponds to the lower bound on the expected optimality of each prior $\log p_{\pi_i}(O = 1)$, with O denoting the event of achieving optimal return and $R(\cdot)$ denotes the return. We refer the reader to [Abdolmaleki et al., 2018, Salter* et al., 2022] for proofs, assumptions, and further discussion on this lower bound. During transfer using RL, we assume no access to the expert or its demonstrations, that is, $i \in I_{\text{RL}} = \{0, u\} \subset I_{\text{BC}} = \{0, u, e\}$.

Hierarchical policies complicate direct KL evaluation, leading to the adoption of an upper bound commonly expressed as [Salter et al., 2020, Tirumala et al., 2019]:

$$\mathrm{D}_{\mathrm{KL}}(\pi(\tau) \parallel \pi_i(\tau)) \leq \sum_t \mathbb{E}_{\pi(\tau)} \Big[\mathrm{D}_{\mathrm{KL}}(\pi^H(\mathbf{z}_t|\mathbf{x}_k) \parallel \pi_i^H(\mathbf{z}_t|\mathbf{x}_k)) + \mathbb{E}_{\pi^H(\mathbf{z}_t|\mathbf{x}_k)} \Big[\mathrm{D}_{\mathrm{KL}}(\pi^L(\mathbf{a}_t|\mathbf{x}_k, \mathbf{z}_t) \parallel \pi_i^L(\mathbf{a}_t|\mathbf{x}_k, \mathbf{z}_t)) \Big] \Big] \tag{B.5}$$

This bound simplifies if modules are shared (i.e., $\pi_i^L = \pi^L$), in which case the second term is cancelled, or non-hierarchical, in which case the first term is cancelled.

To make both $\mathcal{J}_{\mathrm{BC}}$ and $\mathcal{J}_{\mathrm{RL}}$ in equations (B.3) and (B.4) amendable to off-policy training, we introduce importance weighting, thus reducing off-policy bias at the expense of higher variance. Combining all the above with additional individual term-weighting hyperparameters, β_i^z and β_i^a, we attain:

$$\tilde{\mathrm{D}}_{\mathrm{KL}}^{q(\tau)}(\pi(\tau)||\pi_i(\tau)) := \mathbb{E}_{q(\tau)} \left[\sum_t \nu^q[t] \Big(\beta_i^z C_{i,h}(\mathbf{z}_t|\mathbf{x}_k) + \beta_i^a \, \mathbb{E}_{\pi^H(\mathbf{z}_t|\mathbf{x}_k)} \left[C_{i,l}(\mathbf{a}_t|\mathbf{x}_k, \mathbf{z}_t) \right] \Big) \right]$$

$$\text{where } \zeta_i^n = \frac{\mathbb{E}_{\pi^H(z_i|\mathbf{x}_i,k)} \left[\pi^L(\mathbf{a}_i|\mathbf{x}_i, \mathbf{z}_i, k) \right]}{n(\mathbf{a}_i|\mathbf{x}_i, k)},$$

$$\nu^n = \left[\zeta_1^n, \zeta_1^n \zeta_2^n, \ldots, \prod_{i=1}^{\tau_t} \zeta_i^n \right],$$

$$C_{\mu,\epsilon}(y) = \log \left(\frac{\pi_\epsilon^r(y)}{\pi_{\mu,\epsilon}^r(y)} \right)$$

$$-\mathrm{D}_{\mathrm{KL}}(\pi(\tau)||\pi_e(\tau)) \geq - \sum_{i \in \{0,u,e\}} \tilde{\mathrm{D}}_{\mathrm{KL}}^{\pi_e(\tau)}(\pi(\tau)||\pi_i(\tau)) \tag{B.6}$$

$$\mathbb{E}_{\substack{p(\mathcal{K}), \\ \pi_0(\tau)}} [\log(O = 1|\tau, k)] \geq \mathbb{E}_{\pi_b(\tau)} \left[\sum_t \nu^{\pi_b}[t] \cdot r_k(\mathbf{x}_t, \mathbf{a}_t) \right] - \sum_{i \in \{0,u\}} \tilde{\mathrm{D}}_{\mathrm{KL}}^{\pi_b(\tau)}(\pi(\tau)||\pi_i(\tau)) \tag{B.7}$$

Where $\tilde{\mathrm{D}}_{\mathrm{KL}}^{q(\tau)}(\pi(\tau)||\pi_i(\tau))$ (for $\beta_i^z = \beta_i^a = 1$) is an unbiased estimate for the aforementioned upper bound in equation (B.5) using q's experience. ζ_i^n is the importance weight for timestep i, between π and arbitrary policy n. $\nu^n[t]$ is the t^{th} element of ν^n; the cumulative importance weight product at timestep t. Equations (B.6) and (B.7) are the behavioral cloning and reinforcement learning lower bounds used for the policy gradients. We refer the reader to [Salter* et al., 2022]

for the derivation and necessary conditions of these bounds. For behavioral cloning, this bounds the KL-divergence between hierarchical policy π and expert policy π_e. For reinforcement learning, this bounds the expected optimality for the learned prior policy, π_0. Intuitively, maximizing this particular bound maximizes return for both policy and prior, while minimizing the disparity between them. Regularizing against an uninformative prior, π_u, encourages highly-entropic policies, further aiding in exploration and stabilising learning [Haarnoja et al., 2018d, Igl et al., 2019].

While importance weights could be used for the behavioral cloning, we did not observe benefits of using them and thus omit them for the skill acquisition phase.

We employ module sharing ($\pi_i^L = \pi^L$, unless otherwise stated) and freeze certain modules during distinct phases, and thus never employ more than two β hyperparameters at any given time, simplifying the hyperparameter optimization. These weights balance an exploration/exploitation trade-off. We use a categorical latent space, explicitly marginalizing over it, rather than using sampling approximations [Jang et al., 2016]. For behavioral cloning, we train for one epoch, that is, on expectation, once over each sample in the replay buffer.

B.2.3 Critic Learning

The lower bound presented in equation (B.7) is non-differentiable due to rewards being sampled from the environment. Therefore, as is common in the RL literature [Lillicrap et al., 2015, Mnih et al., 2015], we approximate the return of policy π with a critic, Q. For sample efficiency, we train in an off-policy manner with TD-learning [Sutton, 1988] using the Retrace algorithm [Munos et al., 2016] to provide a low-variance, low-bias, policy evaluation operator:

$$Q_t^{\text{ret}} := Q'(\mathbf{x}_t, \mathbf{a}_t, k) + \sum_{j=t}^{\infty} \epsilon_j^t \left[r_k(\mathbf{x}_j, \mathbf{a}_j) + \mathbb{E}_{\substack{\pi^H(\mathbf{z}|\mathbf{x}_{j+1},k),\\ \pi^L(\mathbf{a}'|\mathbf{x}_{j+1},\mathbf{z},k)}} \left[Q'(\mathbf{x}_{j+1}, \mathbf{a}', k) \right] - Q'(\mathbf{x}_j, \mathbf{a}_j, k) \right] \tag{B.8}$$

$$\mathcal{L}(Q) = \mathbb{E}_{\substack{p(\mathcal{K}),\\ \pi_b(\tau)}} \left[(Q(\mathbf{x}_t, \mathbf{a}_t, k) - \arg\min_{Q_t^{\text{ret}}} (Q_t^{\text{ret}}))^2 \right] \qquad \epsilon_j^t = \gamma^{j-t} \prod_{i=t+1}^{j} \zeta_i^b \tag{B.9}$$

Where Q_t^{ret} represents the policy return evaluated via Retrace. Q' is the target Q-network, commonly used to stabilize critic learning [Mnih et al., 2015], and is updated periodically with the current Q values. Importance weights are not ignored here, and are clipped between $[0, 1]$ to prevent exploding gradients [Munos et al., 2016]. To further reduce bias and overestimates of our target, Q_t^{ret}, we apply the double Q-learning trick, [Hasselt, 2010], and concurrently learn two target Q-networks, Q'. Our critic is trained to minimize the loss in equation (B.9), which regularizes the critic against the minimum of the two targets produced by both target networks.

B.3 Experimental Details

This section provides further details on our experimental setup, including environments, used architectures, and hyperparameters. We build off of the softlearning code base [Haarnoja et al., 2018b]. Algorithmic details not mentioned in the following sections are omitted and kept constant with the original SAC implementation [Haarnoja et al., 2018b]. For all experiments, we sample batch size number of **entire episodes** of experience during training.

B.3.1 Environments

This section delves deeper into the specifics of the environments used in our experiments, including the expert setups employed for data collection.

`CorridorMaze`

In the `CorridorMaze` environment, the agent's goal is to traverse through a series of corridors, requiring completion of a corridor cycle, i.e. reaching the end and returning to the origin, before proceeding to the next corridor.

State, Action, and Task Spaces For a `CorridorMaze` with N_c corridors, each of length N_l, the underlying state is a tuple $\mathbf{s} = (c, l) \in (\{0, \dots, N_c\}, \{0, \dots, N_l\})$, denoting in which corridor and how deep into it the agent is. The initial state is set to $p(\mathbf{s}_0) = (0, 0)$, such that the agent starts from the origin, i.e. the intersection of

all corridors. Similarly, the task identifier, $k \in \{1, \ldots, N_c\}$, specifies which of the N_c corridors the agent is required to traverse next. The action space is continuous with $\mathbf{a} \in [0, 1]$, directing the agent's movement within or between the corridors.

Transition Function The transition of the agent within the maze is governed by a deterministic function, with separate logic for the origin and the corridor states:

$$f(\mathbf{s}, \mathbf{a}) = \begin{cases} f_0(\mathbf{a}) & \text{if } \mathbf{s} = \mathbf{0} \\ f_1(\mathbf{s}, \mathbf{a}) & \text{if } \mathbf{s} \neq \mathbf{0}, \end{cases}$$

with f_0 handling the origin states and f_1 the transition corridor transitions. Specifically, $f_0(\mathbf{a})$ decides which, if any, corridor the agent enters, while $f_1(\mathbf{s}, \mathbf{a})$ computes the agent's next position within a corridor. These functions are defined as:

$$\begin{aligned} f_0(\mathbf{a}) &= \begin{cases} (0, 0) & \text{if } \frac{w}{2} < a \\ \left(\min\left(\left\lfloor \frac{2\mathbf{a}}{w} \cdot N_c \right\rfloor + 1, N_c\right), 1\right) & \text{else,} \end{cases} \\ f_1(\mathbf{s}, \mathbf{a}) &= (c_{\text{new}}, l_{\text{new}}), \\ \text{with } l_{\text{new}} &= \begin{cases} l - 1 & \text{if } \mathbf{a} \in [\frac{c-1}{N_c}, \frac{c-1+\frac{w}{2}}{N_c}] \text{ and } 0 < l, \\ l + 1 & \text{if } \mathbf{a} \in [\frac{c-\frac{w}{2}}{N_c}, \frac{c}{N_c}] \text{ and } l < N_l, \\ l & \text{else,} \end{cases} \\ \text{and } c_{\text{new}} &= \begin{cases} 0 & \text{if } l_{\text{new}} = 0 \\ c & \text{else} \end{cases} \end{aligned}$$

The width parameter w controls the range of actions that lead to a transition, influencing the task's exploration difficulty. Unless otherwise stated, we use $w = 0.9$ for all experiments.

Observation In practice, the agent receives a scaled one-hot encoding of the current corridor index and depth, translating the conceptual state tuple into the policy input. This vector is of length N_c, with the active corridor's position marked by the agent's depth within it, and all other elements set to 0. That is (with the slight abuse of notation and reuse of the state variable to denote the agent's observation):

$$\mathbf{s} = (s_1, s_2, \ldots, s_{N_c})^T, \quad \text{where } s_i = \begin{cases} l & \text{if } i = c \\ 0 & \text{otherwise} \end{cases}$$

The task identifier is similarly one-hot encoded vector of length N_c, with the target corridor's position marked by 1 and all other elements set to 0.

Reward Function We consider two different variations of the environment, *semi-sparse* and *sparse*, each providing a different level of reward sparsity. In the semi-sparse variation, $r_k^{\text{semi-sparse}}$ awards a reward of 1 for two scenarios: when the agent reaches the end of the designated corridor (*half-corridor cycle*) and when it successfully returns to the origin after completing the corridor traversal. For all other actions, a reward of 0 is given. Conversely, in the sparse variation, r_k^{sparse} returns 1 only upon the successful completion of all the corridor cycles in the task, making it a more challenging variation due to the increased sparsity of rewards.

Expert Setup The expert samples actions uniformly within the optimal action range for each state, as defined by the transition function above, ensuring corridor traversal in the correct sequence.

Stack

Our `CubeStack` domain modifies the 7-degree-of-freedom FetchPickAndPlace-v0 robotic environment [Plappert et al., 2018] available in the OpenAI Gym suite [Brockman et al., 2016] to introduce a more complex and nuanced cube stacking task. We make the following modifications to the original environment.

1. We introduce additional cubes of varying colors and masses, along with a designated goal pad for stacking.

2. The cube spawn locations are predetermined and set equidistant around the goal pad, as depicted in figure 4.2.

3. To reduce episode lengths, the number of control sub-steps was increased from 20 to 60.

4. A transparent hollow tube was placed around the goal, to simplify the stacking task by preventing the collapse of stacked objects due to structural instabilities.

5. The robot arm is consistently spawned above the goal pad to standardize the starting position for each episode.

Observation, Action, and Task Spaces The action space of the robot is 4-dimensional, with the first three dimensions of the action representing the desired Cartesian displacement along the xyz-axes of the end effector and the final dimension for controlling the closing and opening of the gripper.

We modify the observation space to include the gripper position and grasp state, as well as the object's global positions and relative positions with respect to the arm. Velocities are excluded from the observation.

The task identifier, k, is a one-hot vector specifying the next block to be stacked.

Reward Function The reward function, r_k^{sparse}, grants a reward of 1 for placing the correct object on top of the cube stack and 0 otherwise.

Expert Setup The expert is set up to stack the blocks in the given order. Its strategy is broken down into six sequential segments:

1. *Positioning Above Block:* Move the gripper to a location 20 cm above the target block, keeping it open.

2. *Lowering to Block:* Vertically lower the gripper to 5 mm above the block without closing.

3. *Grasping:* Close the gripper until the object is securely grasped.

4. *Lifting:* Vertically raise the gripper back to 20cm above the initial position, maintaining grip on the block.

5. *Positioning Above Pad:* Relocate the gripper above the target pad while holding the block.

6. *Releasing:* Open the gripper to drop the block onto the target pad, completing the stacking action.

We use MuJoCo's [Todorov et al., 2012] inverse kinematics functions and PD controller with action gains set to 21 to produce desired actions. The transition between segments is triggered upon reaching the specified target locations. When opening or closing the gripper in states 3 and 6, we apply actions of $+0.05$ and -0.1 to the gripped dimension. For stage 3, we continue closing the gripper until there are contact forces between the gripper and the cube. For stage 6, we continue opening the gripper until the block has dropped such that it is within 14 cm of the target pad. To prevent fully deterministic samples, which can be problematic for behavioral cloning, we inject noise into the expert actions. Specifically, we add Gaussian noise with a diagonal covariance and standard deviation of 0.2 per dimension. We do not apply noise to the gripper closing or opening actions.

B.3.2 Model Architectures

We continue by outlining the shared model architectures across domains and experiments. Each policy network ($\pi^H, \pi^L, \pi_0^H, \pi_0^L$) uses a feedforward module outlined in table B.1. Where necessary, the policy outputs are squashed with tanh function to match the predefined action ranges. The critic is similarly parameterized as a feedforward network but is not hierarchical. To handle historical inputs, we tile and flatten the inputs to form a 1-dimensional input array. Where necessary, we ensure that the input always remains of fixed size by appropriately left-padding zeros. For π^H and π_0^H we use a categorical latent space of size 10, which we found sufficient to express the diversity of behaviors exhibited in our domains. Table B.2 presents an overview of each setup for all methods from the main text, including inputs to each module, level at which KL-regularization occurs ($\mathbf{z}$ or $\mathbf{a}$), which modules are shared, and which modules are reused across sequential tasks. None of the reused modules are given access to task-dependent information, namely task identifier k, or exteroceptive information, e.g. cube locations in the `CubeStack` domain. This choice ensures reused modules generalize across the task instances.

Table B.1: Feedforward Module, $\pi_{(0)}^{\{H,L\}}$

hidden layers	$(512, 512)$
hidden layer activation	relu
output activation	linear

Table B.2: Full experimental setup. Describes inputs to each module, level over which KL-regularization occurs, which modules are shared, and which are reused across training and transfer tasks.

Name	Learn $\mathbf{m}$	Module Input				KL Level	$\pi^L = \pi_0^L$	Reuse
		π_0^H	π_0^L	π^L	π^H			
APES	✓	$\mathbf{x}_{t-20:t}$	$\mathbf{s}_t$	$\mathbf{s}_t$	$\mathbf{x}_k$	$\mathbf{z}$	✓	π_0^H, π_0^L, π^L
APES$^{\text{H20}}$	✗	$\mathbf{x}_{t-20:t}$	$\mathbf{s}_t$	$\mathbf{s}_t$	$\mathbf{x}_k$	$\mathbf{z}$	✓	π_0^H, π_0^L, π^L
APES$^{\text{H10}}$	✗	$\mathbf{x}_{t-10:t}$	$\mathbf{s}_t$	$\mathbf{s}_t$	$\mathbf{x}_k$	$\mathbf{z}$	✓	π_0^H, π_0^L, π^L
APES$^{\text{H1}}$	✗	$\mathbf{x}_{t-1:t}$	$\mathbf{s}_t$	$\mathbf{s}_t$	$\mathbf{x}_k$	$\mathbf{z}$	✓	π_0^H, π_0^L, π^L
APES$^{\text{S}}$	✗	$\mathbf{s}_t$	$\mathbf{s}_t$	$\mathbf{s}_t$	$\mathbf{x}_k$	$\mathbf{z}$	✓	π_0^H, π_0^L, π^L
APES$^{\text{NO-PRIOR}}$	-	-	-	$\mathbf{s}_t$	$\mathbf{x}_k$	-	-	π^L
SAC$^{\text{HIER,REC}}$	-	-	-	$\mathbf{s}_t$	$\mathbf{x}_k$	-	-	-
SAC$^{\text{REC}}$	-	-	-	$\mathbf{x}_k$	-	-	-	-
APES$^{\text{H1,KL-A}}$	✗	$\mathbf{x}_{t-1:t}$	$\mathbf{s}_t$	$\mathbf{s}_t$	$\mathbf{x}_k$	$\mathbf{a}$	✗	π_0^H, π_0^L
APES$^{\text{H1,FLAT}}$	✗	-	$\mathbf{x}_{t-1:t}$	$\mathbf{s}_t$	$\mathbf{x}_k$	$\mathbf{a}$	✗	π_0^L

B.3.3 Behavioral Cloning

For the behavioral cloning setup, we use a noisy expert controller to create experience to learn from. We apply DAgger [Ross et al., 2011] during data collection and training of policy π as we found that this resulted in a higher success rate in solving tasks. The noise levels were chosen to be small enough so that the expert still managed to complete the task. We trained our policies for one epoch, i.e. in expectation once over each collected data sample. It may be possible to be more sample efficient by increasing the ratio of gradient steps to data collection, but we did not explore this direction.

Table B.3 presents the final hyperparameters for the BC phase. We performed hyperparameter sweeps mainly for the β-parameters, weighting different KL term components (equation (B.6)). In our setting, these β parameters have a total of two degrees of freedom. As we stop gradients flowing from π_0 to π, choice of β_0 does not affect the interplay of individual loss terms (as long as it is non-zero),

Table B.3: Behavioral Cloning Hyperparameters.

Environment	`CorridorMaze`	`CubeStack`
$\pi_{(i)}$ learning rate	3e−4	3e−4
$\mathbf{z}$ categorical size	10	10
π^H history length	24	5
β_u^z	1e−3	1e−3
β_u^a	1e−2	1e−2
α_m	1e−1	1e−1
DAGGER rate	1e−1	1e−1
Batch size	128	128
Episode length	24	35

so we set it to 1. Similarly, since only the relative magnitude of β_e^a to β_u^z and β_u^a matters, we also set $\beta_e^a = 1$. For the remaining two β hyperparameters, β_u^z and β_u^a, we performed a hyperparameter sweep over three orders of magnitude, that is, three values across each dimension resulting in a grid of nine parameter pairs, to obtain the reported values. In practice π_0 consists of multiple trained priors $\pi_0 = \{\pi_i\}_{i\in\{0,\dots,N\}}$, all sharing the same β hyperparameters.

These β values were first optimized for the fixed APES ablations *without* learned attention. For the learned attention version of APES, we picked the best-performing hyperparameters (as listed in table B.3) and additionally optimized for α_m (the hyperparameter in equation (4.4) weighting π_0^H mask's sparsity term relative to the rest of RL and BC objectives). For these values we ran a similar sweep over three orders of magnitude ($\alpha_m \in \{1\text{e}0, 1\text{e}{-1}, 1\text{e}{-2}\}$).

The final hyperparameters were chosen from these sweeps based on the resulting $D_{KL}(\pi^H||\pi_0^H)$ and $\rho(\mathbf{m})$ values. For methods with fixed attention, we choose the parameters minimizing $D_{KL}(\pi^H||\pi_0^H)$, while for the attention, we manually choose the parameters while preferring $\rho(\mathbf{m})$ without significantly sacrificing performance. Each result was run over four trials with independently sampled seeds. We observed only small variations in learning across trials and used the models from the best-performing trial for transfer. We performed small sweeps for learning rates, $\mathbf{z}$ size, DAGGER rate, and batch size during the development of the algorithm. These

values had only minor impact on the learning performance and were fixed to the listed values throughout the reported trials.

We found that, for both BC and RL setups (described in the next section), conditioning π^H on the entire history sometimes degraded the learning performance, while providing no benefit. Our final results use π^H history length of 24 for `CorridorMaze` and 5 for `CubeStack` environments during the behavioral cloning phase. These same lengths are also used for both π^H and Q in the reinforcement learning setup described in the next section.

We prevent gradient flow from π_0 to π to ensure fairer comparison between ablations so that each prior distills knowledge from the same high-performing policy π and dataset. Training multiple pairs of π and π_0 jointly could lead to different learned priors influencing the quality of each policy π, confounding comparisons. Our focus is on how priors influence knowledge distillation and transfer rather than on modifying π itself. We also confirm that KL-distillation losses for the priors converge consistently across seeds and tasks, ensuring a fair comparison.

B.3.4 Reinforcement Learning

Table B.4: Reinforcement Learning Hyperparameters.

Environment	`CorridorMaze`		`CubeStack`
Transfer task	$N_c = 2$	$N_c = 4$	$N_c = 4$
Reward Type	sparse	semi-sparse	sparse
Q learning rate	3e−6	3e−5	3e−5
π learning rate	3e−4	3e−4	3e−4
$\beta_0^{z/a}$	1e−2	1e−1	5e−2
$\beta_u^{z/a}$	1e−2	0	5e−4
Q update rate	6e−4	6e−4	6e−4
Retrace λ	0.99	0.99	0.99
Batch size	128	128	128
Episode length	30	60	65

During the reinforcement learning stage in training, we freeze the prior and low-level policy for the applicable model variants. All reused modules transferred across

sequential tasks are frozen and any modules that are not shared, such as π^H for most experiments, are initialized randomly across tasks. Depending on the ablation study, our regularization targets either the latent or action space. This applies whether our models are hierarchical, share low-level policies, or incorporate pre-trained components (including the low-level policy and prior). Consequently, regularization is applied using at most two β hyperparameters and the hyperparameter sweeps follow the same procedure as in the behavioral cloning phase outlined above. We did not sweep over Retrace λ, batch size, or episode length. The learning rates were swept for values between 3e$-$7 and 3e$-$3. The final hyperparameters were chosen to maximize the transfer performance. For Retrace, we clip the importance weights between $[0, 1]$, as is done by Munos et al. [2016], and perform λ-returns rather than n-step returns. We found the Retrace operator important in improving the sample-efficiency of learning.

B.4 APES Policy Rollouts

This subsection investigates the converged APES$^{\text{H1}}$ policy's behaviors in `CorridorMaze` and `CubeStack` transfer domains, with a focus on understanding the behavior through categorical probability distributions of $\pi^H(\mathbf{x}_k)$ and $\pi_0^H(IGF(\mathbf{x}_k))$.

For the `CorridorMaze` tasks, as illustrated in figures B.2 and B.3, APES$^{\text{H1}}$ effectively navigates the corridors in the correct sequence, thus successfully solving the task. The categorical distributions for these domains remain relatively entropic. In general, the latent categories cluster into those that lead the agent deeper down a corridor and those that return the agent back to the hallway. The policy and prior distributions align closely, except at the corridor intersections where the prior π_0 is multi-modal — as is desired for transferability for an unknown number or ordering of the hallways — but the policy π needs to deviate from the multi-modal prior and act deterministically given the task information in order to traverse the optimal corridor. For the `CorridorMaze`'s hallway, as shown in figure B.2, the prior allocates one category to each of the four corridors. Such behavior would not be possible with a non-hierarchical flat prior.

In the `CubeStack 4 blocks` domain, presented in figure B.4, APES$^{\text{H1}}$ demonstrates its ability to correctly stack the cubes according to their masses. The categorical latent space exhibits similar trends as for the `CorridorMaze` domain. Most noteworthy is the behavior of both the policy and prior at the bottleneck state, in this case the drop zone above the block stack where the cubes are to be placed. This location is visited five times within the episode: at the start $\mathbf{s}_0$, and four more times upon stacking each block. Interestingly, for this state, the prior becomes increasingly less entropic upon each successive visit. This suggests that the prior has learned that the number of feasible high-level actions, corresponding to which block to stack next, reduces upon each visit, as there remain fewer lighter blocks to stack. It is also interesting that for $\mathbf{s}_0$, the red categorical value is more favored than the rest. Here, the red categorical value corresponds to moving towards cube 0, the heaviest cube. This behavior is as expected, as during the behavioral cloning phase, this cube was stacked first more often than the others, given its mass. For this domain, akin to `CorridorMaze`, the policy deviates most from the prior at the bottleneck state, as here it needs to behave deterministically to optimally choose which cube to stack next, whereas the prior is agnostic to this.

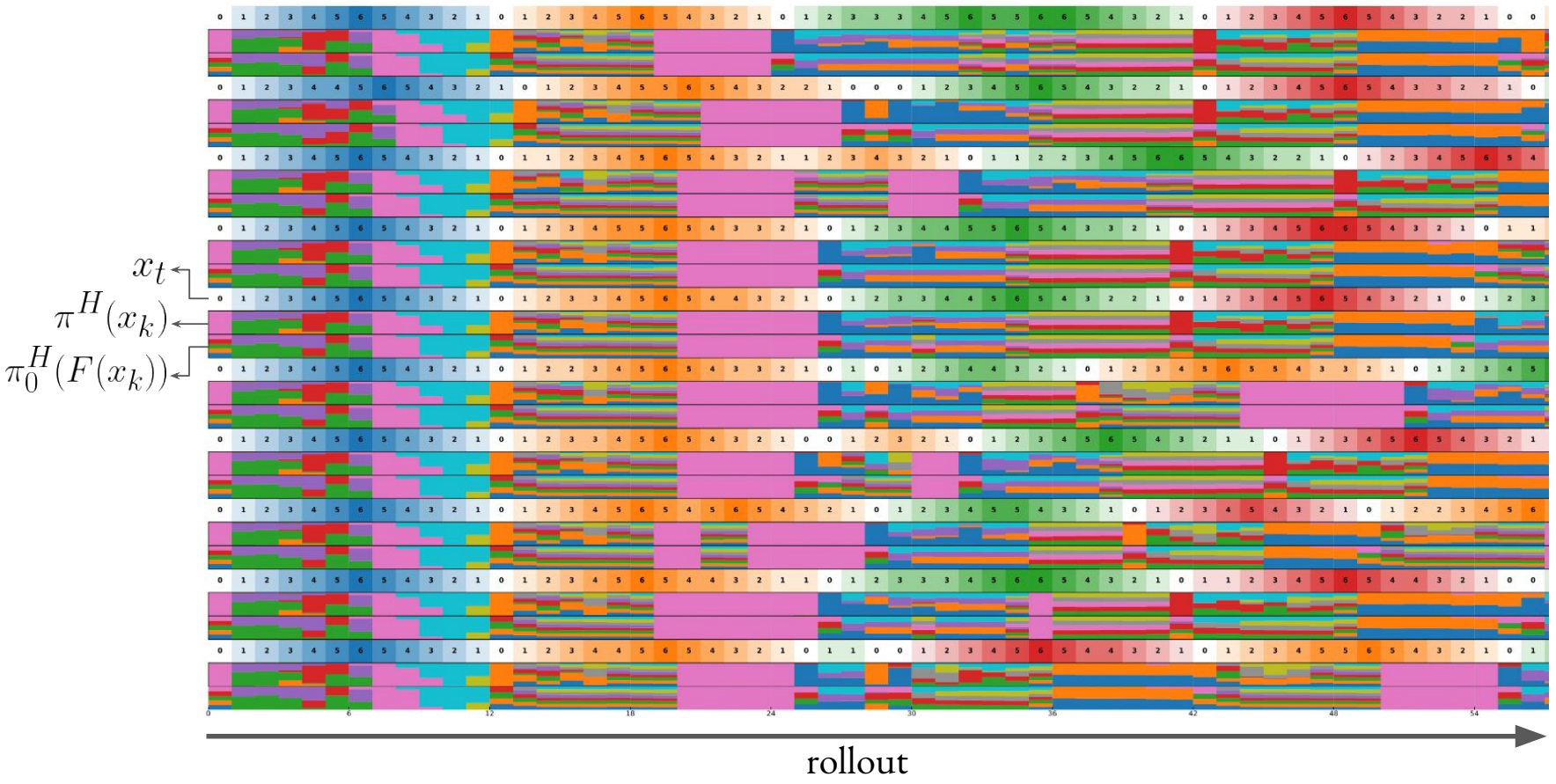


Figure B.2: `CorridorMaze` $N_c = 4$: Ten rollouts for the converged APES$^{\text{H1}}$ policy. The episodes are stacked vertically and the time steps throughout the episode rollout horizontally. The task involves navigating through blue and orange corridors. The color on the top row for each episode, labeled as $\mathbf{x}_t$ represents the corridor currently traversed and the overlaid number and color intensity the depth in that corridor. Similarly, the categorical latent space distributions for policy π^H and prior π_0^H are shown as histograms for each time step, with the category's height denoting the probability of the colored category (histogram colors are independent of the corridor label colors). The policy deviates significantly from the prior at the hallway (white state with label 0), which is a bottleneck state where the prior is multi-modal but the policy requires determinism to solve the task. The information gating function $IGF(\cdot)$ is denoted as $F(\cdot)$ for brevity.

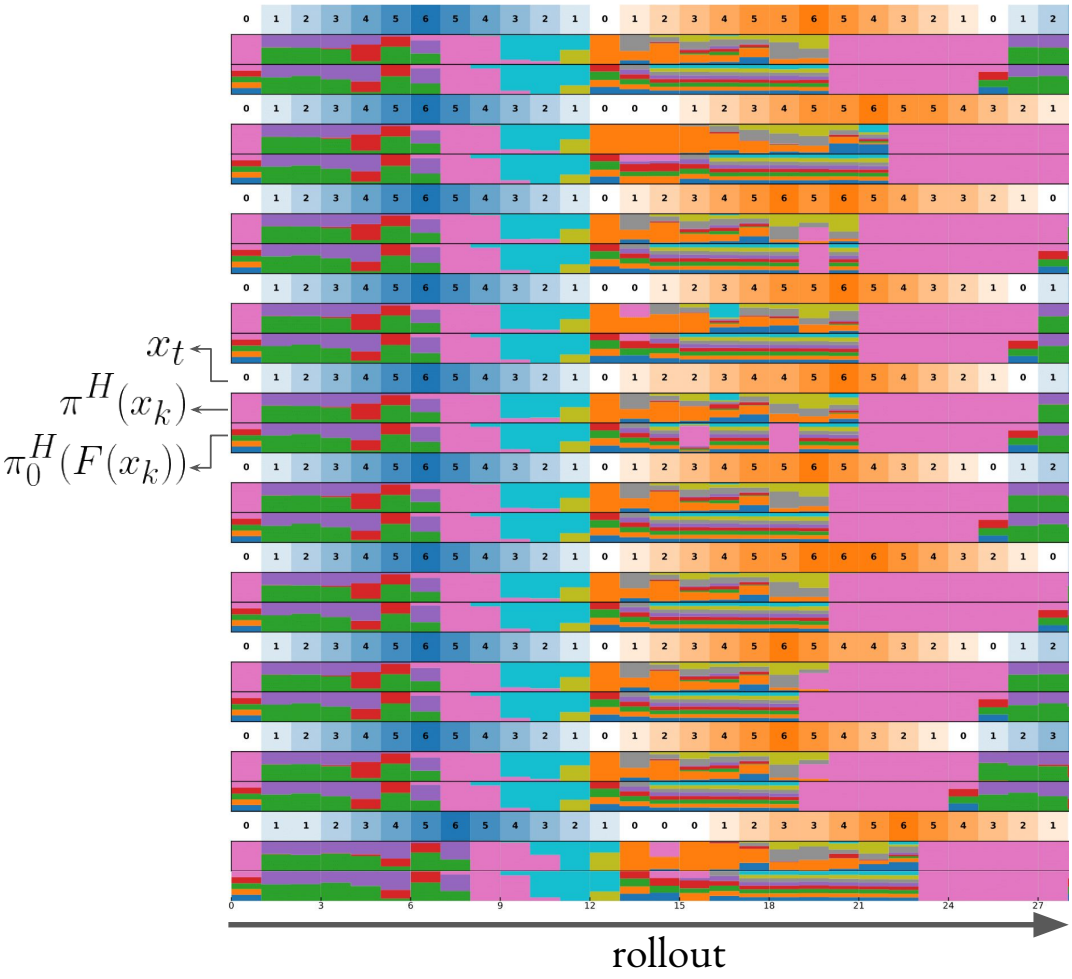


Figure B.3: `CorridorMaze` $N_c = 2$: Ten rollouts for the converged APES$^{\text{H1}}$ policy. See figure B.2 for details on the visualization.

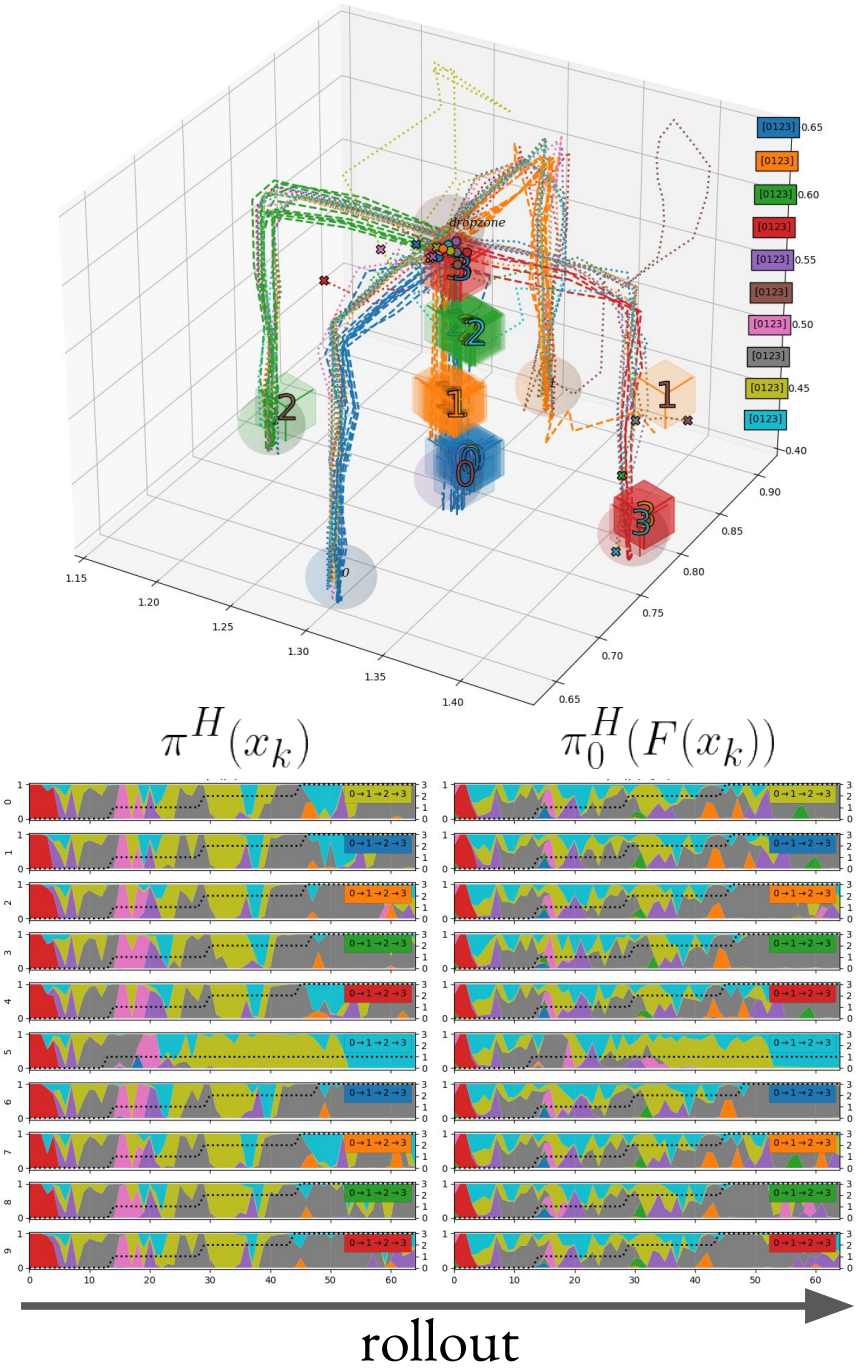


Figure B.4: `CubeStack` $N_c = 4$: (Top) Ten policy rollouts for the converged APES$^{\text{H1}}$ policy. The task involves stacking the cubes in sequence $(0, 1, 2, 3)$. (Bottom) Corresponding policy distributions, similar to figure B.3, are shown for each time step in the episode. Horizontal dashed lines indicate the current stacking sub-task, transitioning upward with each successful cube placement. The policy deviates significantly from the prior at the cube drop zone, which is a bottleneck state where the prior is multi-modal but the policy requires determinism to solve the task. The information gating function $IGF(\cdot)$ is denoted as $F(\cdot)$ for brevity.

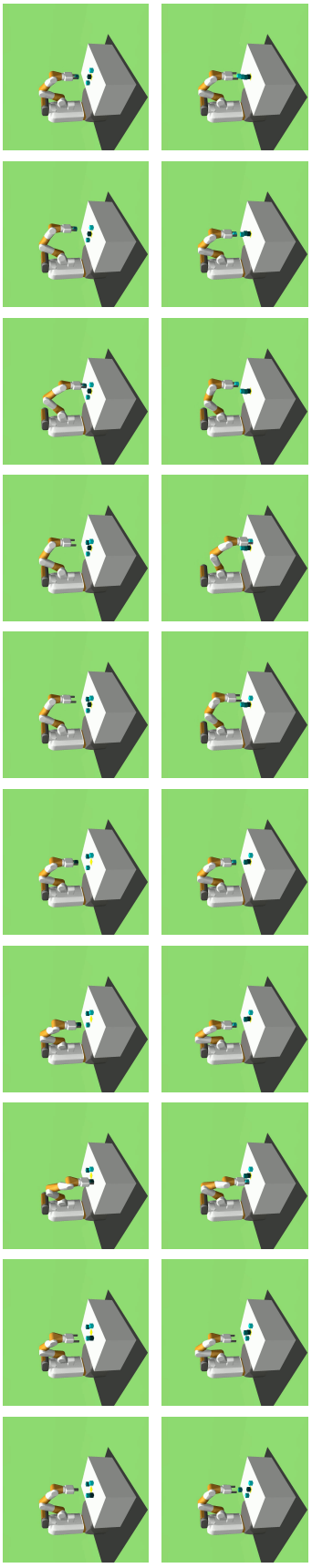


Figure B.5: Typical policy rollout, early during training, for APES on `CubeStack` 4 blocks domain. The still frames of the rollout unroll from left to right and top to bottom. APES explores at the individual block stacking level.

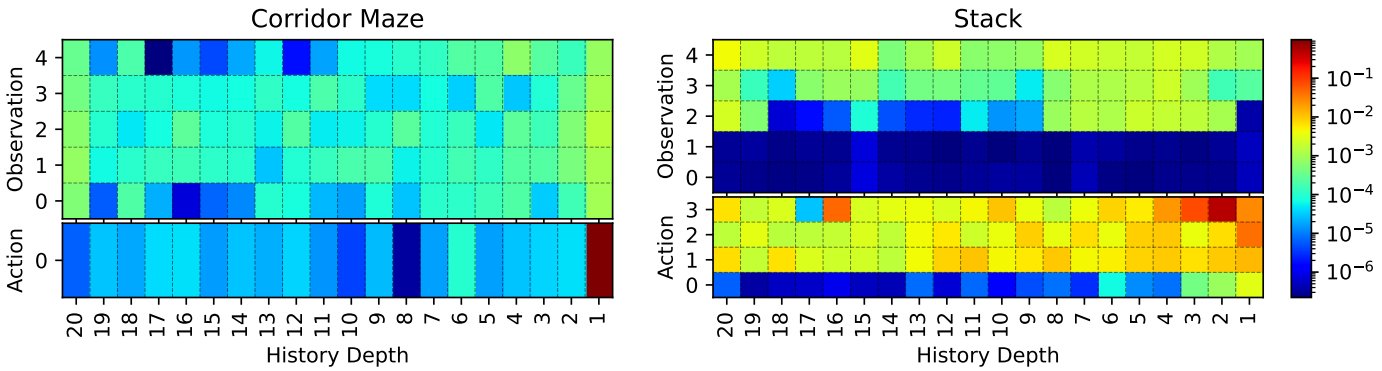


Figure B.6: *Soft* attention distribution of the APES π_0^H, plotted as $\log_{10}(\mathbf{m}_\mathbf{w})$ for the `CorridorMaze` (Left) and `CubeStack` (Right) domains. Color coding shown on the right, red indicating high and blue low attention values.

B.5 APES with Soft Attention Masks

In an earlier version of the APES work, we investigated the use of soft, continuous attention masks, such that $\mathbf{m}_\mathbf{w} \in [0,1]^{\dim \mathbf{x}_k}$. In this setting, we interpret each mask $\mathbf{m}_\mathbf{w}$ as a probability distribution by applying softmax transformation over $\mathbf{w}$, $\mathbf{m}_\mathbf{w} = \text{softmax}(\mathbf{w})$, and define the attention penalty as the entropy over the mask:

$$\mathcal{L}_{\text{ATTENTION}}(\mathbf{m}) \coloneqq \mathcal{H}(\mathbf{m})$$

The entropy penalty $\mathcal{H}(\mathbf{m})$ drives the model towards sparse, low-entropy attention configurations similar to the ℓ_1-penalty in equation (4.5) for the hard mask case.

We note that this soft attention does not fully eliminate dimensions in the same way that hard attention does, thus losing the strict connection with theorems theorems 4.3.1 and 4.3.2. In practice, however, it is observed to often lead to many 0-attention elements [Mott et al., 2019, Salter et al., 2020], effectively reducing the information processed and maintaining focus on relevant inputs. Our findings, which we present below, support these findings.

We provide analysis similar to the one in section 4.5.3, except this time with the hard masks replaced with soft masks.

In figure B.6, we present full attention maps for APES, presenting both intra-observation and intra-action attention mechanisms. For the `CorridorMaze` domain, the model predominantly focuses on the most recent action $\mathbf{a}_t$. This observation

implies that, for a vast majority of environmental states, except when at corridor ends or intersections, the previous action is indicative enough to determine the optimal next step, i.e., to continue traversing along the corridor. Consequently, APES has learned its attention mechanism to reflect this strategy, allocating more attention to where it is most useful, while still paying some attention to broader observations when it needs to navigate complex environmental states.

We highlight here that, despite the observation dimensions taking very low attention values, the soft masking does not preclude π_0^H from using the information from those dimensions, as the model can learn to simply rescale the low-magnitude values if needed. Indeed, the model appears to use some of this information, for example in the hallway intersection and ends in the `CorridorMaze` task.

The overall trend in the `CubeStack` domain follows a similar pattern as with the hard mask: the model primarily allocates its attention to a recent history of actions, with attention for actions decaying continuously towards the past. The attention assigned to actions related to gripper manipulation (in the bottom row of figure B.6) diminishes more rapidly than for other actions. Again, this finding reflects the functional logic of stacking tasks, where successive gripper actions possess high but quickly diminishing correlation. Furthermore, APES largely disregards observations related to the gripper's position, indicating that the model effectively infers this information from other observation space aspects and recent action history. These selective attention patterns suggest that the soft attention may be a viable practical alternative to the hard masking, despite its disconnect from the theory.

C

Interactive Full-Body Motion Imitation with Model-Predictive Control

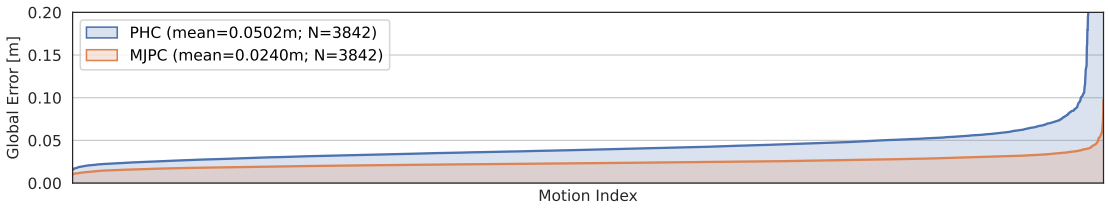
C.1 Additional Tracking Error Figures

Luo et al. [2021, 2023a] report tracking errors both for local and global tracking errors. For completeness, we provide the global tracking errors in figure C.1. The global tracking errors follow a similar trend to the local errors presented in figure 5.4 and discussed in section 5.5.2.

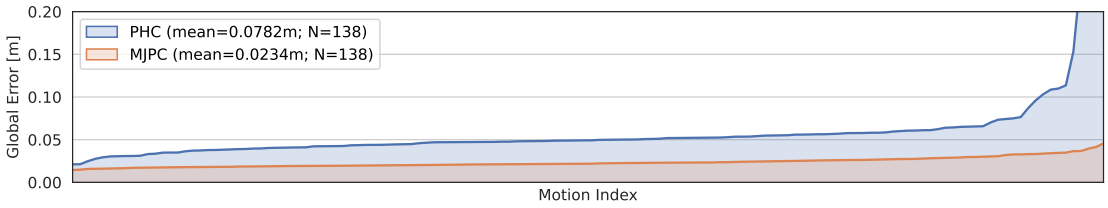
C.2 SMPLHumanoid Modifications

As discussed in section 5.5.1, we modify the `SMPLHumanoid`'s joints and actuators to stabilize the model for MuJoCo. The original model is obtained (in MuJoCo model format) from the official codebase provided by Luo et al. [2023a][1] We implement three key modifications to the model: First, for each joint, we reduce the *stiffness* parameter by a factor of 40 and the *damping* parameter by a factor of 10. Second, we substitute the model's original motor actuators with filtered position actuators (section 5.4). Finally, we remove 20 redundant joints — toe xy-axes, neck x-axis, head y-axis, thorax y-axis, elbow xy-axes, wrist y-axis, and hand xyz-axes — to

[1] `https://github.com/ZhengyiLuo/PHC/blob/79ec27653988d3d4b51ff63772aef0753b6ed230/phc/data/assets/mjcf/smpl_humanoid.xml`



(a) $E^{\text{mpjpe-g}}$ for 3842 randomly chosen sequences from AMASS test split.



(b) $E^{\text{mpjpe-g}}$ for the 138 AMASS test sequences.

Figure C.1: Per-sequence tracking errors for the keyframe motions from AMASS splits from Luo et al. [2023a]. Each point on the x-axis presents a motion in the dataset and the y-axis presents global tracking error, $E^{\text{mpjpe-g}}$ (lower is better), for the training set (a, top) and the test set (b, bottom). For better readability, the motions within each method are shown in ascending order based on the corresponding tracking error. The average error over the motions for each split is shown in the corresponding legend.

reduce the model's degrees of freedom to enable faster planning. We observe no change in tracking quality, either visually or in the tracking errors, from omitting the joints, but are able to run the planner about 30% faster.

C.3 Model and Planner Parameters

We use a value of $\theta^{dynamics} = 0.03$ for both the `SimpleHumanoid` and `SMPLHumanoid` models. The gain and bias parameters for each joint, θ^{gain} and θ^{bias}, are defined as:

$$\theta^{gain} = k_p * \text{slope}$$
$$\theta_0^{bias} = k_p * (q_{\min} - \text{slope} * u_{\min})$$
$$\theta_1^{bias} = -k_p$$

where k_p is the proportional gain parameter and slope is defined as $\text{slope} = \frac{(q_{\max} - q_{\min})}{(u_{\max} - u_{\min})}$. The $q_{\min}$ and $q_{\max}$ denote the maximum and minimum positions of the joint and $u_{\min}$ and $u_{\max}$ denote the control range corresponding to that joint. In practice, we set $[u_{\min}, u_{\max}] = [-1, +1]$. Finally, the force range of the actuators (`forcerange` parameter in MuJoCo) is limited to the range $[f_{\min}, f_{\max}]$.

Table C.1: Joint and actuator parameters for the `SimpleHumanoid` model. All the joints in the model are 1-dimensional hinge joints but we have grouped the sides and axes with matching parameters within each joint. The degrees of freedom (DOFs) column indicates the number of joints under the grouping.

			joint	joint parameters				actuator parameters		
name	side	axis	DOFs	stiffness	damping	$q_{\min}$	$q_{\max}$	k_p	$f_{\min}$	$f_{\max}$
abdomen	—	y	1	10	15	$-\frac{5\pi}{12}$	$+\frac{\pi}{6}$	300	-300	+300
abdomen	—	x	1	10	20	$-\frac{7\pi}{36}$	$+\frac{7\pi}{36}$	200	-200	+200
abdomen	—	z	1	20	20	$-\frac{\pi}{4}$	$+\frac{\pi}{4}$	180	-180	+180
ankle	left,right	x	2	3	3	$-\frac{5\pi}{18}$	$+\frac{5\pi}{18}$	50	-50	+50
ankle	left,right	y	2	6	6	$-\frac{5\pi}{18}$	$+\frac{5\pi}{18}$	120	-120	+120
elbow	left,right	—	2	0	5	$-\frac{5\pi}{9}$	$+\frac{5\pi}{18}$	90	-90	+90
hip	left,right	z	2	10	10	$-\frac{\pi}{3}$	$+\frac{7\pi}{36}$	200	-200	+200
hip	left,right	x	2	10	10	$-\frac{\pi}{6}$	$+\frac{\pi}{18}$	200	-200	+200
hip	left,right	y	2	10	15	$-\frac{5\pi}{6}$	$+\frac{\pi}{9}$	300	-300	+300
knee	left,right	—	2	1	8	$-\frac{8\pi}{9}$	$+\frac{\pi}{90}$	160	-160	+160
shoulder1	left,right	—	2	1	6	$-\frac{2\pi}{3}$	$+\frac{\pi}{3}$	120	-120	+120
shoulder2	left,right	—	2	1	6	$-\frac{2\pi}{3}$	$+\frac{\pi}{3}$	120	-120	+120

The actuator and joint parameters for `SimpleHumanoid` are listed in table C.1 and for `SMPLHumanoid` in table C.2.

Table C.2: Joint and actuator parameters for the `SMPLHumanoid` model. All the joints in the model are 1-dimensional hinge joints but we have grouped the sides and axes with matching parameters within each joint. The degrees of freedom (DOFs) column indicates the number of joints under the grouping.

			joint	joint parameters				actuator parameters		
name	side	axis	DOFs	stiffness	damping	$q_{\min}$	$q_{\max}$	k_p	$f_{\min}$	$f_{\max}$
Ankle	left,right	x,z	4	20	8	$-\frac{\pi}{4}$	$+\frac{\pi}{4}$	250	-250	+250
Ankle	left,right	y	2	20	8	$-\frac{\pi}{2}$	$+\frac{\pi}{2}$	250	-250	+250
Chest	—	x,y,z	3	20	10	$-\frac{\pi}{3}$	$+\frac{\pi}{3}$	250	-250	+250
Elbow	left	z	1	20	5	$-\pi$	0	250	-250	+250
Elbow	right	z	1	20	5	0	$+\pi$	250	-250	+250
Head	—	y,z	2	20	5	$-\frac{\pi}{2}$	$+\frac{\pi}{2}$	250	-250	+250
Hip	left,right	x,y,z	6	20	8	$-\frac{\pi}{2}$	$+\frac{\pi}{2}$	250	-250	+250
Knee	left,right	x,z	4	20	8	$-\frac{\pi}{32}$	$+\frac{\pi}{32}$	250	-250	+250
Knee	left,right	y	2	20	8	0	$+\pi$	250	-250	+250
Neck	—	y,z	2	20	5	$-\frac{\pi}{2}$	$+\frac{\pi}{2}$	250	-250	+250
Shoulder	left,right	x,y	4	20	5	$-\pi$	$+\pi$	250	-250	+250
Shoulder	left,right	z	2	20	5	-4π	$+4\pi$	250	-250	+250
Spine	—	x,y,z	3	20	10	$-\frac{\pi}{3}$	$+\frac{\pi}{3}$	250	-250	+250
Thorax	left,right	x,z	4	20	5	$-\frac{\pi}{32}$	$+\frac{\pi}{32}$	250	-250	+250
Toe	left,right	y	2	20	5	$-\pi$	$+\pi$	250	-250	+250
Torso	—	x,y,z	3	20	10	$-\frac{\pi}{3}$	$+\frac{\pi}{3}$	250	-250	+250
Wrist	left,right	x,z	4	20	3	$-\pi$	$+\pi$	250	-250	+250

References

A. Abdolmaleki, J. T. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018. 4.1, 4.2.2, B.2.2

J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 1

H. Ahn, T. Ha, Y. Choi, H. Yoo, and S. Oh. Text2action: Generative adversarial synthesis from language to action. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5915–5920. IEEE, 2018. 5.2.1

C. Ahuja and L.-P. Morency. Language2pose: Natural language grounded pose forecasting. In *2019 International Conference on 3D Vision (3DV)*, pages 719–728. IEEE, 2019. 5.2.1

A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *arXiv preprint arXiv:2010.13611*, 2020. 4.1, 4.2.3, 4.4, 4.6

S. Amin, M. Gomrokchi, H. Satija, H. van Hoof, and D. Precup. A survey of exploration methods in reinforcement learning. *arXiv preprint arXiv:2109.00157*, 2021. 2.2.5

A. Antos, R. Munos, and C. Szepesvári. Fitted q-iteration in continuous action-space mdps. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NIPS'07, page 9–16, Red Hook, NY, USA, 2007. Curran Associates Inc. ISBN 9781605603520. 3.5.1

A. Antos, C. Szepesvári, and R. Munos. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 71(1):89–129, 2008. doi: 10.1007/s10994-007-5038-2. 3.5.1

K. Arulkumaran, A. Cully, and J. Togelius. Alphastar: An evolutionary computation perspective. In *Proceedings of the genetic and evolutionary computation conference companion*, pages 314–315, 2019. 1

J. Asmuth and M. Littman. Learning is planning: near bayes-optimal reinforcement learning via monte-carlo tree search. *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 19–26, 01 2011. 3.2.2

N. I. Badler, C. B. Phillips, and B. L. Webber. *Simulating humans: computer graphics animation and control*. Oxford University Press, 1993. 5.2.1

M. Bagatella, S. Christen, and O. Hilliges. Sfp: State-free priors for exploration in off-policy reinforcement learning. *Transactions on Machine Learning Research*, 2022. 4.1, 4.1, 4.2.3, 4.3, 1, 4.4, 4.5.2, 4.2, 4.5.3, 4.5.3, 4.6

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 1

L. Baird. Residual algorithms: Reinforcement learning with function approximation. *Machine Learning-International Workshop Then Conference-*, pages 30–37, July 1995. ISSN 00043702. doi: 10.1.1.48.3256. 3.3.2

J. F. Bard. Some properties of the bilevel programming problem. *J. Optim. Theory Appl.*, 68(2):371–378, February 1991. ISSN 0022-3239. 3.4.1

E. Barsoum, J. Kender, and Z. Liu. Hp-gan: Probabilistic 3d human motion prediction via gan. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 1418–1427, 2018. 5.2.1

M. J. Beal. *Variational algorithms for approximate Bayesian inference*. PhD thesis, Gatsby Computational Neuroscience Unit, University College London, 2003. 3.2.4

M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 449–458, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. 3.2.4

R. Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957. 2.2.4

Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013. 4.4

K. Bergamin, S. Clavet, D. Holden, and J. R. Forbes. Drecon: data-driven responsive control of physics-based characters. *ACM Transactions On Graphics (TOG)*, 38 (6):1–11, 2019. 5.2.2

C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019. 1

S. Bhatnagar, D. Precup, D. Silver, R. S. Sutton, H. R. Maei, and C. Szepesvári. Convergent temporal-difference learning with arbitrary smooth function approximation. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1204–1212, 2009. 3.3.2, 3.6.1, 3.6.1, A.2.1, A.2.1, A.2.1

S. Bohez, S. Tunyasuvunakool, P. Brakel, F. Sadeghi, L. Hasenclever, Y. Tassa, E. Parisotto, J. Humplik, T. Haarnoja, R. Hafner, et al. Imitate and repurpose: Learning reusable robot movement skills from human and animal behaviors. *arXiv preprint arXiv:2203.17138*, 2022. 5.1, 5.2.2, 5.2.2

M. M. Botvinick, Y. Niv, and A. G. Barto. Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *cognition*, 113(3): 262–280, 2009. 4.2.2

J. A. Boyan and A. W. Moore. Generalization in reinforcement learning: Safely approximating the value function. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 369–376. MIT Press, 1995. 3.6.1, A.2.2, A.2.3

D. Brandfonbrener and J. Bruna. Geometric insights into the convergence of nonlinear td learning. In *ICLR 2020*, 2019. 3.5.1

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016. 3.6.2, 3.6.2, 4.5.1, 1, A.3, A.3.1, A.3.3, B.3.1

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 1

J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. *Advances in neural information processing systems*, 31, 2018. 2.2.5

S. R. Buss. Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation*, 17(1-19):16, 2004. 5.4

C. Chen, W. Xie, W. Huang, Y. Rong, X. Ding, Y. Huang, T. Xu, and J. Huang. Progressive feature alignment for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 627–636, 2019. 5.2.2

M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. 1

N. Chentanez, M. Müller, M. Macklin, V. Makoviychuk, and S. Jeschke. Physics-based motion capture imitation with deep reinforcement learning. In *Proceedings of the 11th ACM SIGGRAPH Conference on Motion, Interaction and Games*, pages 1–10, 2018. 5.1, 5.2.2

K. Ciosek, Q. Vuong, R. Loftin, and K. Hofmann. Better exploration with optimistic actor critic. *Advances in Neural Information Processing Systems*, 32, 2019. 2.2.5, 3.2.3, 3.6.2, A.3

K. Ciosek, V. Fortuin, R. Tomioka, K. Hofmann, and R. Turner. Conservative uncertainty estimation by fitting prior networks. In *Eighth International Conference on Learning Representations*, April 2020. 3.4.2, 3.6.2

CMU Graphics Lab. Carnegie-mellon motion capture database, 2003. URL `http://mocap.cs.cmu.edu`. Funded by NSF EIA-0196217. 5.2.2

D. Cox and D. Hinkley. *Theoretical statistics*. Chapman and Hall, London, 1974. ISBN 0412124203. 3.1

M. da Silva, Y. Abe, and J. Popović. Interactive simulation of stylized human locomotion. *ACM Transactions on Graphics (TOG)*, 27(3):1–10, 2008. 5.2.2

M. Da Silva, Y. Abe, and J. Popović. Simulation of human motion data using short-horizon model-predictive control. In *Computer Graphics Forum*, volume 27, pages 371–380. Wiley Online Library, 2008. 5.2.2

C. Dann, G. Neumann, and J. Peters. Policy evaluation with temporal differences: A survey and comparison. *Journal of Machine Learning Research*, 15:809–883, 2014. 3.6.1, A.2.2, A.2.2, A.2.2

B. de Finetti. La prévision : ses lois logiques, ses sources subjectives. *Annales de l'institut Henri Poincaré*, 7(1):1–68, 1937. 3.1

R. Dearden, N. Friedman, S. Russell, et al. Bayesian q-learning. *Aaai/iaai*, 1998: 761–768, 1998. 3.5.1

N. Dilokthanakul, P. A. Mediano, M. Garnelo, M. C. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016. 5.2.1

S. S. Du, J. Chen, L. Li, L. Xiao, and D. Zhou. Stochastic variance reduction methods for policy evaluation. In D. Precup and Y. W. Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1049–1058, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. 3.6.1

Y. Duan, T. Shi, Z. Zou, Y. Lin, Z. Qian, B. Zhang, and Y. Yuan. Single-shot motion completion with transformer. *arXiv preprint arXiv:2103.00776*, 2021. 5.2.1

M. O. Duff and A. Barto. *Optimal Learning: Computational Procedures for Bayes-Adaptive Markov Decision Processes*. PhD thesis, University of Massachusetts Amherst, 2002. AAI3039353. 3.2.1

J. Engel, L. Hantrakul, C. Gu, and A. Roberts. Ddsp: Differentiable digital signal processing. *arXiv preprint arXiv:2001.04643*, 2020. 1

Y. Engel, S. Mannor, and R. Meir. Bayes meets bellman: The gaussian process approach to temporal difference learning. In *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, page 154–161, 2003. ISBN 1577351894. 3.5.1

Y. Engel, S. Mannor, and R. Meir. Reinforcement learning with gaussian processes. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 201–208, New York, NY, USA, 2005. Association for Computing Machinery. ISBN 1595931805. doi: 10.1145/1102351.1102377. URL `https://doi.org/10.1145/1102351.1102377`. 3.5.1

A. Escontrela, X. B. Peng, W. Yu, T. Zhang, A. Iscen, K. Goldberg, and P. Abbeel. Adversarial motion priors make good substitutes for complex reward functions. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 25–32. IEEE, 2022. 5.2.2

F. Farshidian, M. Neunert, and J. Buchli. Learning of closed-loop motion control. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1441–1446. IEEE, 2014. 2.2.5

V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018. 2.2.5

M. Fellows. *Bayesian and Vriational Inference for Reinforcement Learning*. PhD thesis, University of Oxford, 2021. 2.1.5, 3.2.2, 3.3.1

M. Fellows, K. Hartikainen, and S. Whiteson. Bayesian bellman operators. *Advances in Neural Information Processing Systems*, 34:13641–13656, 2021. 1.1, 3, 3.3.2, 3.3.2, 3.3.3, 3.6.1

M. Fellows, B. Kaplowitz, C. S. De Witt, and S. Whiteson. Bayesian exploration networks. *arXiv preprint arXiv:2308.13049*, 2023. 6.0.1

Y. Feng, L. Li, and Q. Liu. A kernel loss for solving the bellman equation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 15456–15467. Curran Associates, Inc., 2019. 3.5.1

M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, G. Alexander, M. Vlad, M. Remi, H. Demis, P. Olivier, B. Charles, and L. Shane. Noisy networks for exploration. In *Proceedings of the International Conference on Representation Learning (ICLR 2018)*, Vancouver (Canada), 2018. 3.5.1

Z. Fu, X. Cheng, and D. Pathak. Deep whole-body control: learning a unified policy for manipulation and locomotion. In *Conference on Robot Learning*, pages 138–149. PMLR, 2023. 5.2.2

L. Fussell, K. Bergamin, and D. Holden. Supertrack: Motion tracking for physically simulated characters using supervised learning. *ACM Transactions on Graphics (TOG)*, 40(6):1–13, 2021. 5.1, 5.2.2, 5.2.2

Y. Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016. 3.2.4, 3.3.3

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1050–1059. JMLR.org, 2016. 3.5.1

A. Galashov, S. M. Jayakumar, L. Hasenclever, D. Tirumala, J. Schwarz, G. Desjardins, W. M. Czarnecki, Y. W. Teh, R. Pascanu, and N. Heess. Information asymmetry in kl-regularized rl. *arXiv preprint arXiv:1905.01240*, 2019. 4.1, 4.2.3, 4.3, 4.3.1, 4.4, 4.5.2, 4.2, 4.5.3, 4.6

D. M. Gavrila. The visual analysis of human movement: A survey. *Computer vision and image understanding*, 73(1):82–98, 1999. 5.2.1

J. Gehring, G. Synnaeve, A. Krause, and N. Usunier. Hierarchical skills for efficient exploration. *Advances in Neural Information Processing Systems*, 34:11553–11564, 2021. 4.6

M. Ghavamzadeh, S. Mannor, J. Pineau, and A. Tamar. Bayesian reinforcement learning: A survey. *arXiv preprint arXiv:1609.04436*, 2016. 3.1, 3.2.1, 3.2.4

K. Gong, B. Li, J. Zhang, T. Wang, J. Huang, M. B. Mi, J. Feng, and X. Wang. Posetriplet: Co-evolving 3d human pose estimation, imitation, and hallucination under self-supervision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11017–11027, 2022. 5.2.2

A. Goyal, R. Islam, D. Strouse, Z. Ahmed, M. Botvinick, H. Larochelle, Y. Bengio, and S. Levine. Infobot: Transfer and exploration via the information bottleneck. *arXiv preprint arXiv:1901.10902*, 2019. 4.1, 4.1, 4.2.3, 4.3, 4.6

A. Guez, D. Silver, and P. Dayan. Scalable and efficient bayes-adaptive reinforcement learning based on monte-carlo tree search. *Journal of Artificial Intelligence Research*, 48:841–883, 10 2013. doi: 10.1613/jair.4117. 3.2.2

C. Guo, X. Zuo, S. Wang, S. Zou, Q. Sun, A. Deng, M. Gong, and L. Cheng. Action2motion: Conditioned generation of 3d human motions. In *Proceedings of the 28th ACM International Conference on Multimedia*, pages 2021–2029, 2020. 5.2.1

T. Haarnoja, K. Hartikainen, P. Abbeel, and S. Levine. Latent space policies for hierarchical reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018a. 4.1, 4.2.2

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning (ICML)*, 2018b. 3.1, 3.6.2, 4.1, 4.4, 4.2, A.3, A.3.3, B.3

T. Haarnoja, A. Zhou, S. Ha, J. Tan, G. Tucker, and S. Levine. Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*, 2018c. 4.3

T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018d. 3.5, 3.6.2, 4.5.2, A.3.1, B.2.2

P. Hämäläinen, S. Eriksson, E. Tanskanen, V. Kyrki, and J. Lehtinen. Online motion synthesis using sequential monte carlo. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014. 5.2.2

P. Hämäläinen, J. Rajamäki, and C. K. Liu. Online control of simulated humanoids using particle belief propagation. *ACM Transactions on Graphics (TOG)*, 34(4): 1–13, 2015. 5.2.2

D. Han, H. Eom, J. Noh, and J. S. Shin. Data-guided model predictive control based on smoothed contact dynamics. In *Proceedings of the 37th Annual Conference of the European Association for Computer Graphics*, pages 533–543, 2016. 5.2.2, 5.6

J. Hao, T. Yang, H. Tang, C. Bai, J. Liu, Z. Meng, P. Liu, and Z. Wang. Exploration in deep reinforcement learning: From single-agent to multiagent domain. *IEEE Transactions on Neural Networks and Learning Systems*, 2023. 2.2.5

J. Harb, P.-L. Bacon, M. Klissarov, and D. Precup. When waiting is not an option: Learning options with a deliberation cost. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 4.6

F. G. Harvey and C. Pal. Recurrent transition networks for character locomotion. In *SIGGRAPH Asia 2018 Technical Briefs*, SA '18, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450360623. doi: 10.1145/3283254.3283277. URL `https://doi.org/10.1145/3283254.3283277`. 5.2.1

F. G. Harvey, M. Yurick, D. Nowrouzezahrai, and C. Pal. Robust motion in-betweening. *ACM Trans. Graph.*, 39(4), aug 2020. ISSN 0730-0301. doi: 10.1145/3386569.3392480. URL `https://doi.org/10.1145/3386569.3392480`. 5.2.1, 5.2.2

L. Hasenclever, F. Pardo, R. Hadsell, N. Heess, and J. Merel. Comic: Complementary task learning & mimicry for reusable skills. In *International Conference on Machine Learning*, pages 4105–4115. PMLR, 2020. 5.1, 5.2.2, 5.2.2, 5.4

H. V. Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2613–2621, 2010. 4.4, B.2.3

K. Hausman, J. T. Springenberg, Z. Wang, N. Heess, and M. Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018. 4.1, 4.6

N. Heess, G. Wayne, D. Silver, T. Lillicrap, T. Erez, and Y. Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2944–2952, 2015. 2.2.5

R. Howard. *Dynamic programming and Markov processes*. Technology Press of Massachusetts Institute of Technology, 1960. 2.2.4

T. Howell, N. Gileadi, S. Tunyasuvunakool, K. Zakka, T. Erez, and Y. Tassa. Predictive sampling: Real-time behaviour synthesis with mujoco. *arXiv preprint arXiv:2212.00541*, 2022. 5.1, 1, 5.4, 5.5.1

M. Igl, A. Gambardella, J. He, N. Nardelli, N. Siddharth, W. Böhmer, and S. Whiteson. Multitask soft option learning. *arXiv preprint arXiv:1904.01033*, 2019. 4.6, B.2.2

D. Jacobson and D. Mayne. *Differential Dynamic Programming*. Modern analytic and computational methods in science and mathematics. American Elsevier Publishing Company, 1970. ISBN 9780444000705. 5.4

E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016. 4.4, B.2.2

B. Jiang, X. Chen, W. Liu, J. Yu, G. Yu, and T. Chen. Motiongpt: Human motion as a foreign language. *arXiv preprint arXiv:2306.14795*, 2023. 5.2.1

M. I. Jordan. *Learning in Graphical Models*. MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-60032-3. 3.2.4

L. P. Kaelbling. Associative reinforcement learning: Functions in k-dnf. *Machine Learning*, 15:279–298, 1994. 2.2.5, 3.2.3

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998. 2.2.6

G. Kalweit and J. Boedecker. Uncertainty-driven imagination for continuous deep reinforcement learning. In *Conference on robot learning*, pages 195–206. PMLR, 2017. 2.2.5

A. Kamat and D. Precup. Diversity-enriched option-critic. *arXiv preprint arXiv:2011.02565*, 2020. 4.6

H. J. Kappen, V. Gómez, and M. Opper. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012. 4.2.1

T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 1

M. Kaufmann, E. Aksan, J. Song, F. Pece, R. Ziegler, and O. Hilliges. Convolutional autoencoders for human motion infilling. In *2020 International Conference on 3D Vision (3DV)*, pages 918–927. IEEE, 2020. 5.2.1

K. Khetarpal, M. Klissarov, M. Chevalier-Boisvert, P.-L. Bacon, and D. Precup. Options of interest: Temporal abstraction with interest functions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4444–4451, 2020a. 4.4.1, 4.6

K. Khetarpal, M. Riemer, I. Rish, and D. Precup. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*, 2020b. 4.6

J. Kim, J. Kim, and S. Choi. Flame: Free-form language-based motion synthesis & editing. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence*, pages 8255–8263, 2023. 5.2.1

D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference for Learning Presentations (ICLR)*, 2015. A.2.2

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In Y. Bengio and Y. LeCun, editors, *ICLR*, 2014. 3.2.4

J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017. 4.4.1

J. Z. Kolter and A. Y. Ng. Near-bayesian exploration in polynomial time. In *Proceedings of the 26th annual international conference on machine learning*, pages 513–520, 2009. 2.2.5, 3.2.3

T. Komura, I. Habibie, D. Holden, J. Schwarz, and J. Yearsley. A recurrent variational autoencoder for human motion synthesis. In *The 28th British Machine Vision Conference*, 2017. 5.2.1

V. Konda and J. Tsitsiklis. Actor-critic algorithms. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12, pages 1008–1014. MIT Press, 2000. 3.5

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6402–6413. Curran Associates, Inc., 2017. 3.2.4

T. Lampe and M. Riedmiller. Approximate model-assisted neural fitted q-iteration. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 2698–2704. IEEE, 2014. 2.2.5

H.-Y. Lee, X. Yang, M.-Y. Liu, T.-C. Wang, Y.-D. Lu, M.-H. Yang, and J. Kautz. Dancing to music. *Advances in neural information processing systems*, 32, 2019. 5.2.1

Y. Lee, S. Kim, and J. Lee. Data-driven biped control. *ACM Trans. Graph.*, 29 (4), jul 2010. ISSN 0730-0301. doi: 10.1145/1778765.1781155. URL `https://doi.org/10.1145/1778765.1781155`. 5.1, 5.2.2

S. Levine and V. Koltun. Guided policy search. In *International Conference on Machine Learning (ICML)*, pages 1–9, 2013. 2.2.5

S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. 4.4.1, 4.6

J. Li, Y. Yin, H. Chu, Y. Zhou, T. Wang, S. Fidler, and H. Li. Learning to generate diverse dance motions with transformer. *arXiv preprint arXiv:2008.08171*, 2020. 5.2.1

W. Li and E. Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *First International Conference on Informatics in Control, Automation and Robotics*, volume 2, pages 222–229. SciTePress, 2004. 1

T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. 4.1, B.2.3

X. Lin and M. R. Amer. Human motion modeling using dvgans. *arXiv preprint arXiv:1804.10652*, 2018. 5.2.1

Z. Lipton, X. Li, J. Gao, L. Li, F. Ahmed, and l. Deng. Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems. *AAAI*, 11 2018. 3.5.1

L. Liu, K. Yin, M. van de Panne, T. Shao, and W. Xu. Sampling-based contact-rich motion control. *ACM Trans. Graph.*, 29(4), jul 2010. ISSN 0730-0301. doi: 10.1145/1778765.1778865. URL `https://doi.org/10.1145/1778765.1778865`. 5.1, 5.2.2

L. Liu, K. Yin, and B. Guo. Improving sampling-based motion control. *Comput. Graph. Forum*, 34(2):415–423, may 2015. ISSN 0167-7055. doi: 10.1111/cgf.12571. URL `https://doi.org/10.1111/cgf.12571`. 5.2.2

L. Liu, M. V. D. Panne, and K. Yin. Guided learning of control graphs for physics-based characters. *ACM Transactions on Graphics (TOG)*, 35(3):1–14, 2016. 5.1

S. Liu, G. Lever, Z. Wang, J. Merel, S. A. Eslami, D. Hennes, W. M. Czarnecki, Y. Tassa, S. Omidshafiei, A. Abdolmaleki, et al. From motor control to team play in simulated humanoid football. *Science Robotics*, 7(69):eabo0235, 2022. 4.1, 4.1, 4.2.3, 4.4, 4.6

Y. Luo, H. Xu, Y. Li, Y. Tian, T. Darrell, and T. Ma. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. In *International Conference on Learning Representations*, 2018. 2.2.5

Z. Luo, R. Hachiuma, Y. Yuan, and K. Kitani. Dynamics-regulated kinematic policy for egocentric pose estimation. *Advances in Neural Information Processing Systems*, 34:25019–25032, 2021. 5.1, 5.2.2, 5.5, 5.5.1, 5.5.1, 5.5.1, 5.5.2, 5.5.2, C.1

Z. Luo, J. Cao, K. Kitani, W. Xu, et al. Perpetual humanoid control for real-time simulated avatars. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10895–10904, 2023a. 5.1, 5.2.2, 5.4, 5.4, 5.4, 5.5, 5.5.1, 5.5.1, 5.5.1, 5.5.1, 5.5.1, 5.5.2, 5.5.2, C.1, C.2, C.1

Z. Luo, J. Cao, J. Merel, A. Winkler, J. Huang, K. Kitani, and W. Xu. Universal humanoid motion representations for physics-based control. *arXiv preprint arXiv:2310.04582*, 2023b. 5.2.2, 5.4, 5.4

C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016. 4.4, 2

S. Maheshwari, D. Gupta, and R. K. Sarvadevabhatla. Mugl: Large scale multi person conditional action generation with locomotion. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 257–265, 2022. 5.2.1

N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5442–5451, 2019. 5.2.2, 5.5.1

V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, et al. Isaac gym: High performance gpu-based physics simulation for robot learning. *arXiv preprint arXiv:2108.10470*, 2021. 5.2.2

D. Mensah, N. H. Kim, M. Aittala, S. Laine, and J. Lehtinen. A hybrid generator architecture for controllable face synthesis. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 1–10, 2023. 1

J. Merel, Y. Tassa, D. TB, S. Srinivasan, J. Lemmon, Z. Wang, G. Wayne, and N. Heess. Learning human behaviors from motion capture by adversarial imitation. *arXiv preprint arXiv:1707.02201*, 2017. 5.1, 5.2.2

J. Merel, A. Ahuja, V. Pham, S. Tunyasuvunakool, S. Liu, D. Tirumala, N. Heess, and G. Wayne. Hierarchical visuomotor control of humanoids. *arXiv preprint arXiv:1811.09656*, 2018a. 5.2.2

J. Merel, L. Hasenclever, A. Galashov, A. Ahuja, V. Pham, G. Wayne, Y. W. Teh, and N. Heess. Neural probabilistic motor primitives for humanoid control. *arXiv preprint arXiv:1811.11711*, 2018b. 4.6, 5.1, 5.2.2, 5.4

J. Merel, S. Tunyasuvunakool, A. Ahuja, Y. Tassa, L. Hasenclever, V. Pham, T. Erez, G. Wayne, and N. Heess. Catch & carry: reusable neural controllers for vision-guided whole-body tasks. *ACM Transactions on Graphics (TOG)*, 39(4): 39–1, 2020. 4.1, 4.6, 5.2.2

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 3.4.1, 3.4.2, 3.5, 4.1, B.2.3, B.2.3

A. Mott, D. Zoran, M. Chrzanowski, D. Wierstra, and D. Jimenez Rezende. Towards interpretable reinforcement learning using attention augmented agents. *Advances in Neural Information Processing Systems*, 32, 2019. B.5

U. Muico, Y. Lee, J. Popović, and Z. Popović. Contact-aware nonlinear control of dynamic characters. *ACM Trans. Graph.*, 28(3), jul 2009. ISSN 0730-0301. doi: 10.1145/1531326.1531387. URL `https://doi.org/10.1145/1531326.1531387`. 5.1, 5.2.2

R. Munos, T. Stepleton, A. Harutyunyan, and M. G. Bellemare. Safe and efficient off-policy reinforcement learning. *arXiv preprint arXiv:1606.02647*, 2016. 4.4, B.2.3, B.2.3, B.3.4

K. P. Murphy. *Machine Learning: A Probabilistic Perspective*, chapter 7. The MIT Press, 2012. ISBN 0262018020, 9780262018029. 3.3.3

O. Nachum, H. Lee, S. Gu, and S. Levine. Data-Efficient Hierarchical Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 2018-Decem, pages 3303–3313, 2018. URL `https://sites.google.com/view/efficient-hrl`. 4.6

A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018. 2.2.5

D. H. Nguyen and B. Widrow. Neural networks for self-learning control systems. *IEEE Control systems magazine*, 10(3):18–23, 1990. 2.2.5

B. O'Donoghue, I. Osband, R. Munos, and V. Mnih. The uncertainty Bellman equation and exploration. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3839–3848, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. 3.5.1

A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016. 1

J. O'Rourke and N. I. Badler. Model-based image analysis of human motion using constraint propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-2(6):522–536, 1980. doi: 10.1109/TPAMI.1980.6447699. 5.2.1

I. Osband and B. Van Roy. Why is posterior sampling better than optimism for reinforcement learning? In *International conference on machine learning*, pages 2701–2710. PMLR, 2017. 3.2.3

I. Osband, D. Russo, and B. Van Roy. (more) efficient reinforcement learning via posterior sampling. *Advances in Neural Information Processing Systems*, 26, 2013. 2.2.5, 3.2.3

I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems (NeurIPS)*, pages 4026–4034, 2016. 2.2.5

I. Osband, J. Aslanides, and A. Cassirer. Randomized prior functions for deep reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8617–8629. Curran Associates, Inc., 2018. 2.1.5, 3.1, 3.2.4, 3.4.2, 3.5, 3.5.1, 3.6.2

I. Osband, Y. Doron, M. Hessel, J. Aslanides, E. Sezener, A. Saraiva, K. McKinney, T. Lattimore, C. Szepesvari, S. Singh, et al. Behaviour suite for reinforcement learning. *arXiv preprint arXiv:1908.03568*, 2019a. 2.2.5

I. Osband, B. V. Roy, D. J. Russo, and Z. Wen. Deep exploration via randomized value functions. *Journal of Machine Learning Research*, 20(124):1–62, 2019b. 3.4.2, 3.5, 3.5.1, A.3.2

G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. 4.1, 4.6

H. Park, R. Yu, and J. Lee. Multi-segment foot modeling for human animation. In *Proceedings of the 11th ACM SIGGRAPH Conference on Motion, Interaction and Games*, pages 1–10, 2018. 5.2.2

S. Park, H. Ryu, S. Lee, S. Lee, and J. Lee. Learning predict-and-simulate policies from unorganized human motion data. *ACM Transactions on Graphics (TOG)*, 38(6):1–11, 2019. 5.2.2, 5.2.2

D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017. 3.6.2

T. Pearce, M. Zaki, A. Brintrup, and A. Neely. Uncertainty in neural networks: Bayesian ensembling. *ArXiv Preprint*, abs/1810.05546, 10 2019. 2.1.5, 3.4.2

X. B. Peng, G. Berseth, K. Yin, and M. Van De Panne. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 36(4):1–13, 2017. 5.1, 5.2.2

X. B. Peng, P. Abbeel, S. Levine, and M. Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018a. 5.1, 5.2.2, 5.2.2, 5.4, 5.4

X. B. Peng, A. Kanazawa, S. Toyer, P. Abbeel, and S. Levine. Variational discriminator bottleneck: Improving imitation learning, inverse rl, and gans by constraining information flow. *arXiv preprint arXiv:1810.00821*, 2018b. 5.2.2

X. B. Peng, M. Chang, G. Zhang, P. Abbeel, and S. Levine. Mcp: learning composable hierarchical control with multiplicative compositional policies. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, pages 3686–3697, 2019a. 5.2.2

X. B. Peng, A. Kumar, G. Zhang, and S. Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019b. 4.4.1

X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (ToG)*, 40(4):1–20, 2021. 5.2.2, 5.2.2

X. B. Peng, Y. Guo, L. Halper, S. Levine, and S. Fidler. Ase: Large-scale reusable adversarial skill embeddings for physically simulated characters. *ACM Transactions On Graphics (TOG)*, 41(4):1–17, 2022. 5.1, 5.2.2

K. Pertsch, Y. Lee, and J. J. Lim. Accelerating reinforcement learning with learned skill priors. *arXiv preprint arXiv:2010.11944*, 2020. 4.2.2, 4.1, 4.2.3, 4.3, 4.4, 4.5, 4.5.2, 4.2, 4.5.3, 4.6, B.2.2

K. Pertsch, Y. Lee, Y. Wu, and J. J. Lim. Guided reinforcement learning with learned skills. In *5th Annual Conference on Robot Learning*, 2021. 4.1, 4.1, 4.2.3, 4.6

M. Petrovich, M. J. Black, and G. Varol. Action-conditioned 3d human motion synthesis with transformer vae. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10985–10995, 2021. 5.2.1

M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018. 4.5.1, B.3.1

M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., USA, 1st edition, 1994. ISBN 0471619779. 2.2.3

D. Rao, F. Sadeghi, L. Hasenclever, M. Wulfmeier, M. Zambelli, G. Vezzani, D. Tirumala, Y. Aytar, J. Merel, N. Heess, et al. Learning transferable motor skills with hierarchical latent mixture policies. *arXiv preprint arXiv:2112.05062*, 2021. 4.1, 4.2.3, 4.6

K. Rawlik, M. Toussaint, and S. Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. *Robotics: Science and Systems (RSS)*, 2012. 4.2.1

J. Ren, C. Yu, S. Chen, X. Ma, L. Pan, and Z. Liu. Diffmimic: Efficient motion mimicking with differentiable physics. *arXiv preprint arXiv:2304.03274*, 2023. 5.2.2

A. Reske, J. Carius, Y. Ma, F. Farshidian, and M. Hutter. Imitation learning from mpc for quadrupedal multi-gait control. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5014–5020. IEEE, 2021. 5.2.2

D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015. 3.2.4

M. Riemer, M. Liu, and G. Tesauro. Learning abstract options. *Advances in neural information processing systems*, 31, 2018. 4.6

H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400 – 407, 1951. doi: 10.1214/aoms/1177729586. URL `https://doi.org/10.1214/aoms/1177729586`. 3.3.2

S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011. 5.6, B.2.1, B.3.3

D. Russell, R. Papallas, and M. Dogar. Adaptive approximation of dynamics gradients via interpolation to speed up trajectory optimisation. In *Proceedings of the 2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023. 5.6

S. Salter, D. Rao, M. Wulfmeier, R. Hadsell, and I. Posner. Attention-privileged reinforcement learning. In *Conference on Robot Learning*, 2020. 1.1, 4.6, B.2.2, B.5

S. Salter*, K. Hartikainen*, W. Goodwin, and I. Posner. Priors, hierarchy, and information asymmetry for skill transfer in reinforcement learning. In *The Eleventh International Conference on Learning Representations*, 2022. 4, B.2.2, B.2.2

S. Salter, M. Wulfmeier, D. Tirumala, N. Heess, M. Riedmiller, R. Hadsell, and D. Rao. Mo2: Model-based offline options. *Conference on Lifelong Learning Agents*, 2022. 4.4, 4.4.1, 4.6

J. Schulman, P. Abbeel, and X. Chen. Equivalence between policy gradients and soft Q-learning. *arXiv preprint arXiv:1704.06440*, 2017. 4.2.1

L. X. Shi, J. J. Lim, and Y. Lee. Skill-based model-based reinforcement learning. *arXiv preprint arXiv:2207.07560*, 2022. 4.6

N. Y. Siegel, J. T. Springenberg, F. Berkenkamp, A. Abdolmaleki, M. Neunert, T. Lampe, R. Hafner, N. Heess, and M. Riedmiller. Keep doing what worked: Behavioral modelling priors for offline reinforcement learning. *arXiv preprint arXiv:2002.08396*, 2020. 4.6

H. Sikchi, W. Zhou, and D. Held. Learning off-policy with online planning. In *Conference on Robot Learning*, pages 1622–1633. PMLR, 2022. 4.6

D. Silver, R. S. Sutton, and M. Müller. Sample-based learning and search with permanent and transient memories. In *Proceedings of the 25th international conference on Machine learning*, pages 968–975, 2008. 2.2.5

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017. 1, 4.1

A. Singh, H. Liu, G. Zhou, A. Yu, N. Rhinehart, and S. Levine. Parrot: Data-driven behavioral priors for reinforcement learning. *arXiv preprint arXiv:2011.10024*, 2020. 4.4, 4.6

L. Song, K. Fukumizu, and A. Gretton. Kernel embeddings of conditional distributions: A unified kernel framework for nonparametric inference in graphical models. *IEEE Signal Processing Magazine*, 30(4):98–111, 2013. doi: 10.1109/MSP.2013.2252713. 3.2.2

M. Strens. A bayesian framework for reinforcement learning. In *ICML*, volume 2000, pages 943–950, 2000. 2.2.5, 3.2.3

R. S. Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988. B.2.3

R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier, 1990. 2.2.5

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*, volume 1. MIT press, 2018. 2.2.1, 2.2.4, 2.2.4, 2.2.5

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999a. 2.2.5

R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112 (1-2):181–211, 1999b. 4.6

R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 993–1000, New York, NY, USA, 2009a. ACM. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553501. 3.3.2, 3.6.1

R. S. Sutton, H. R. Maei, and C. Szepesvári. A convergent o(n) temporal-difference algorithm for off-policy learning with linear function approximation. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 1609–1616. Curran Associates, Inc., 2009b. 3.3.2, 3.3.2

B. Tasdighi, M. Haussmann, N. Werge, Y.-S. Wu, and M. Kandemir. Deep exploration with pac-bayes. *arXiv preprint arXiv:2402.03055*, 2024. 3.6.2

Y. Tassa, T. Erez, and E. Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012. 5.2.2, 5.4, 5.4

Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018. 3.6.2, A.3, A.3.2

G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023. 1

Y. W. Teh, V. Bapst, W. M. Czarnecki, J. Quan, J. Kirkpatrick, R. Hadsell, N. Heess, and R. Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 2017-Decem, pages 4497–4507, 2017. 4.1, 4.2.1, 4.2.2, 4.6

G. Tevet, B. Gordon, A. Hertz, A. H. Bermano, and D. Cohen-Or. Motionclip: Exposing human motion generation to clip space. In *European Conference on Computer Vision*, pages 358–374. Springer, 2022a. 5.2.1

G. Tevet, S. Raab, B. Gordon, Y. Shafir, D. Cohen-Or, and A. H. Bermano. Human motion diffusion model. *arXiv preprint arXiv:2209.14916*, 2022b. 5.2.1

W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4):285–294, 12 1933. ISSN 0006-3444. doi: 10.1093/biomet/25.3-4.285. 2.2.5, 3.2.3

S. Thrun. Monte carlo pomdps. *Advances in neural information processing systems*, 12, 1999. 4.5.2

D. Tirumala, H. Noh, A. Galashov, L. Hasenclever, A. Ahuja, G. Wayne, R. Pascanu, Y. W. Teh, and N. Heess. Exploiting hierarchy for learning and transfer in kl-regularized rl. *arXiv preprint arXiv:1903.07438*, 2019. 4.1, 4.2.2, 4.2.2, 4.1, 4.2.3, 4.3, 4.4, 4.4.1, 4.5, 4.5.2, 4.2, 4.6, B.2.2

D. Tirumala, A. Galashov, H. Noh, L. Hasenclever, R. Pascanu, J. Schwarz, G. Desjardins, W. M. Czarnecki, A. Ahuja, Y. W. Teh, et al. Behavior priors for efficient reinforcement learning. *arXiv preprint arXiv:2010.14274*, 2020. 4.1, 4.1, 4.2.3, 4.3, 4.4, 4.4.1, 4.5, 4.5.2, 4.2, 4.5.3

E. Todorov. Linearly-solvable Markov decision problems. In *Advances in Neural Information Processing Systems*, pages 1369–1376. MIT Press, 2007. 4.2.1

E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 5026–5033. IEEE, 2012. 5.1, 5.5.1, B.3.1

A. Touati, H. Satija, J. Romoff, J. Pineau, and P. Vincent. Randomized value functions via multiplicative normalizing flows. In A. Globerson and R. Silva, editors, *UAI*, page 156. AUAI Press, 2019. 3.5.1

J. Tseng, R. Castellon, and K. Liu. Edge: Editable dance generation from music. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 448–458, 2023. 5.2.1

J. N. Tsitsiklis and B. Van Roy. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control*, 42(5):674–690, May 1997. ISSN 2334-3303. doi: 10.1109/9.580874. 3.5.1, 3.6.1, A.2.1, A.2.1

S. Tunyasuvunakool, A. Muldal, Y. Doron, S. Liu, S. Bohez, J. Merel, T. Erez, T. Lillicrap, N. Heess, and Y. Tassa. dm_control: Software and tasks for continuous control. *Software Impacts*, 6:100022, 2020. 5.5.1, 5.5.1

A. W. Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 2000. 2.1.2

N. Vieillard, T. Kozuno, B. Scherrer, O. Pietquin, R. Munos, and M. Geist. Leverage the average: an analysis of regularization in rl. *Advances in Neural Information Processing Systems*, 33, 2020. 3.5

N. Vlassis, M. Ghavamzadeh, S. Mannor, and P. Poupart. Bayesian reinforcement learning. *Reinforcement learning*, pages 359–386, 2012. 3.1, 3.2.4

N. Wagener, A. Kolobov, F. Vieira Frujeri, R. Loynd, C.-A. Cheng, and M. Hausknecht. Mocapact: A multi-task dataset for simulated humanoid control. *Advances in Neural Information Processing Systems*, 35:35418–35431, 2022. 5.1, 5.2.2, 5.2.2

T. Wang, D. Lizotte, M. Bowling, and D. Schuurmans. Bayesian sparse sampling for on-line reward optimization. In *Proceedings of the 22nd international conference on Machine learning*, pages 956–963, 2005. 2.2.5, 3.2.3

T. Wang, Y. Guo, M. Shugrina, and S. Fidler. Unicon: Universal neural controller for physics-based character motion. *arXiv preprint arXiv:2011.15119*, 2020. 5.1, 5.2.2

Z. Wang, J. S. Merel, S. E. Reed, N. de Freitas, G. Wayne, and N. Heess. Robust imitation of diverse behaviors. *Advances in Neural Information Processing Systems*, 30, 2017. 5.2.2

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992. 2.2.5

J. Won, D. Gopinath, and J. Hodgins. A scalable approach to control diverse behaviors for physically simulated characters. *ACM Transactions on Graphics (TOG)*, 39(4):33–1, 2020. 5.1, 5.2.2

Y. Wu, G. Tucker, and O. Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019. 4.6

M. Wulfmeier, A. Abdolmaleki, R. Hafner, J. T. Springenberg, M. Neunert, N. Siegel, T. Hertweck, T. Lampe, N. Heess, and M. Riedmiller. Compositional transfer in hierarchical reinforcement learning. *Robotics: Science and Systems XVI*, 2020a. 4.1, 4.3, 4.4, 4.4.1, 4.5, 4.5.2, 4.2, 4.6

M. Wulfmeier, D. Rao, R. Hafner, T. Lampe, A. Abdolmaleki, T. Hertweck, M. Neunert, D. Tirumala, N. Siegel, N. Heess, et al. Data-efficient hindsight off-policy option learning. *arXiv preprint arXiv:2007.15588*, 2020b. 4.1, 4.6

K. Xie, H. Bharadhwaj, D. Hafner, A. Garg, and F. Shkurti. Latent skill planning for exploration and transfer. *arXiv preprint arXiv:2011.13897*, 2020. 4.6

T. Yang, H. Tang, C. Bai, J. Liu, J. Hao, Z. Meng, P. Liu, and Z. Wang. Exploration in deep reinforcement learning: a comprehensive survey. *arXiv e-prints*, pages arXiv–2109, 2021. 2.2.5

H. Yao, L.-K. Huang, L. Zhang, Y. Wei, L. Tian, J. Zou, J. Huang, et al. Improving generalization in meta-learning via task augmentation. In *International conference on machine learning*, pages 11887–11897. PMLR, 2021. 3.2.2

K. Yin, K. Loken, and M. Van de Panne. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics (TOG)*, 26(3):105–es, 2007. 5.1, 5.2.2

Y. Yuan and K. Kitani. 3d ego-pose estimation via imitation learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 735–750, 2018. 5.2.2

Y. Yuan and K. Kitani. Ego-pose estimation and forecasting as real-time pd control. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10082–10092, 2019. 5.1, 5.2.2

Y. Yuan and K. Kitani. Dlow: Diversifying latent flows for diverse human motion prediction. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pages 346–364. Springer, 2020. 5.2.1

M. Zhang, Z. Cai, L. Pan, F. Hong, X. Guo, L. Yang, and Z. Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. *arXiv preprint arXiv:2208.15001*, 2022. 5.2.1

L. Zintgraf, K. Shiarlis, M. Igl, S. Schulze, Y. Gal, K. Hofmann, and S. Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. *8th International Conference on Learning Representations, ICLR 2020, Virtual Conference, Formerly Addis Ababa ETHIOPIA*, 2020. 3.2.2

L. M. Zintgraf, L. Feng, C. Lu, M. Igl, K. Hartikainen, K. Hofmann, and S. Whiteson. Exploration in approximate hyper-state space for meta reinforcement learning. In *International Conference on Machine Learning*, pages 12991–13001. PMLR, 2021. 3.2.2