Uncertainty Representations in Reinforcement Learning

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
Genehmigte Dissertation von Carlos Enrique Luis Goncalves aus Venezuela
Tag der Einreichung: 07.10.2024, Tag der Prüfung: 06.12.2024

1. Gutachten: Prof. Jan Peters
2. Gutachten: Prof. Marc G. Bellemare
Darmstadt, Technische Universität Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Computer Science Department

Intelligent Autonomous Systems Group

Uncertainty Representations in Reinforcement Learning

Genehmigte Dissertation von Carlos Enrique Luis Goncalves

Tag der Einreichung: 07.10.2024
Tag der Prüfung: 06.12.2024

Darmstadt, Technische Universität Darmstadt

Erklärungen laut Promotionsordnung

§ 8 Abs. 1 lit. c PromO

Ich versichere hiermit, dass die elektronische Version meiner Dissertation mit der schriftlichen Version übereinstimmt.

§ 8 Abs. 1 lit. d PromO

Ich versichere hiermit, dass zu einem vorherigen Zeitpunkt noch keine Promotion versucht wurde. In diesem Fall sind nähere Angaben über Zeitpunkt, Hochschule, Dissertationsthema und Ergebnis dieses Versuchs mitzuteilen.

§ 9 Abs. 1 PromO

Ich versichere hiermit, dass die vorliegende Dissertation selbstständig und nur unter Verwendung der angegebenen Quellen verfasst wurde.

§ 9 Abs. 2 PromO

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 07.10.2024

C. E. Luis Goncalves

Abstract

Reinforcement learning (RL) has achieved tremendous success over the last decade, primarily through massive compute in simulated environments. However, applications of RL in physical systems have lagged behind as a result of open challenges such as sample-efficient learning, partial observability, generalization and adaptation to unseen tasks. Consequently, there exists a large gap to be filled before RL becomes the standard for enabling autonomous systems in the real world.

In this thesis, we argue that proper handling of uncertainty is key to address these challenges. We introduce uncertainty estimation techniques that consider the sequential nature of decision-making, which enable a seamless integration of the resulting uncertainty estimates into RL algorithms.

First, we adopt the model-based RL paradigm and investigate methods that propagate uncertainty from the learned dynamics up to long-term predictions of the *value* of a control policy. Key to these approaches are probabilistic models that separate *aleatoric* and *epistemic* uncertainty: the former is an inherent part of the problem and therefore irreducible, while the latter exists due to a lack of knowledge about the dynamics and can be reduced by strategically collecting more data. We first tackle the problem of estimating the epistemic *variance* around the predicted performance (value) of a policy. We derive a theoretically-grounded estimation algorithm that effectively propagates model uncertainty and recovers the desired variance. We then demonstrate how to use such epistemic variance estimates for improved exploration in tabular problems. For more challenging continuous control tasks, we identify challenges to apply our theory and propose a suitable approximation, which leads to a practical deep RL architecture that accomodates risk-seeking or risk-averse policy optimization. As a natural next step, we show how to efficiently learn an entire *distribution* of policy values, rather than just its mean and variance. The distributional representation of epistemic uncertainty around values is more expressive and allows for a wider range of policy optimization objectives while having low computational overhead. Furthermore, empirical evaluation in diverse control tasks indicate a substantial improvement in final performance and sample-efficiency over state-of-the-art methods.

Next, we consider the problem of partial observability in model-free RL. That is, the environment observations provide limited information for decision-making, therefore the hidden state of the environment must be infered from trajectory data. In this setting, we propose sequence models composed of Kalman filter (KF) layers that perform closed-form Gaussian inference in linear state-space models and train them end-to-end to maximize returns. By design, the KF layers are a drop-in replacement of previous recurrent layers in model-free architectures, but they are equipped with an explicit mechanism for probabilistic filtering of the latent state representation. We empirically demonstrate that KF layers excel in tasks where reasoning over uncertainty is crucial for decision-making.

Zusammenfassung

Das Verstärkungslernen (Reinforcement Learning, RL) hat in den letzten zehn Jahren enorme Erfolge erzielt, insbesondere durch massive Rechenleistung in simulierten Umgebungen. Anwendungen von RL in physischen Systemen sind jedoch aufgrund offener Herausforderungen wie Dateneffizienz, partieller Beobachtbarkeit, und Generalisierung und Anpassung an neue Aufgaben in Verzug geraten. Folglich gibt es noch eine große Lücke, die geschlossen werden muss, bevor RL zum Standard für autonome Systeme in der realen Welt werden kann.

In dieser Arbeit argumentieren wir, dass der richtige Umgang mit Unsicherheit der Schlüssel zur Bewältigung dieser Herausforderungen ist. Wir führen Techniken zur Unsicherheitsabschätzung ein, die die sequentielle Natur der Entscheidungsfindung berücksichtigen und eine nahtlose Integration der resultierenden Unsicherheitsabschätzungen in RL-Algorithmen ermöglichen.

Erstens übernehmen wir das modellbasierte RL-Konzept und untersuchen Methoden, die die Ungewissheit von der gelernten Dynamik in langfristigen Vorhersagen des Wertes eines Reglers propagieren. Das Fundament dieses Ansatzes bilden probabilistische Modelle, die zwischen *aleatorische* und *epistemische* unterscheiden: Erstere ist ein inhärenter Teil des Problems und daher irreduzibel, während letztere aufgrund mangelnder Kenntnisse über die Dynamik besteht und durch strategisches Sammeln von zusätzlichen Daten reduziert werden kann. Als ersten Schritt entwickeln wir eine Methode Methode zur Schätzung der epistemischen *Varianz* um die vorhergesagte Wert eines Reglers. Insbesondere leiten einen theoretisch begründeten Schätzalgorithmus her, der die Modellunsicherheit effektiv propagiert und die gewünschte Varianz bestimmt. Anschließend zeigen wir, wie solche epistemischen Varianzschätzungen für die Exploration in tabellarischen Problemen verbesser kann. Für anspruchsvollere kontinuierliche Steuerungsaufgaben identifizieren wir Herausforderungen bei der Anwendung unserer Theorie und schlagen eine geeignete Annäherung vor, die zu einer praktischen verschachtelten (tiefen) RL-Architektur führt und eine risikofreudige oder risikoaverse Strategieoptimierung ermöglicht. Als natürlichen nächster Schritt wird gezeigt, wie man die gesamte *Verteilung* über Werte effizient lernen kann, und nicht nur deren Mittelwert und Varianz. Die Verteilungsdarstellung der epistemischen Unsicherheit um die Werte herum ist aussagekräftiger und ermöglicht eine breitere Palette von Optimierungszielen bei geringerem Rechenaufwand. Darüber hinaus zeigen empirische Auswertungen in verschiedenen Steuerungsaufgaben eine erhebliche Verbesserung der endgültigen Performanz und der Dateneffizienz im Vergleich zu etablierten Methoden.

Als nächstes betrachten wir das Problem der partiellen Beobachtbarkeit in modellfreiem RL. Das heißt, dass die Beobachtungen der Umgebung nur begrenzte Informationen für die Entscheidungsfindung liefern, weshalb der verborgene Zustand der Umgebung aus den Daten in den Trajektorien abgeleitet werden muss. In dieser Situation schlagen wir Sequenzmodelle vor, die aus Kalman-Filter (KF)-Schichten bestehen, eine Gauß'sche Inferenz in linearen Zustandsraummodellen durchführen und Ende-zu-Ende trainiert werden um die Performance zu maximieren. Die KF-Schichten sind als Drop-in-Ersatz für frühere rekurrente Schichten in modellfreien Architekturen konzipiert, verfügen aber über einen expliziten Mechanismus zur probabilistischen Filterung der latenten Zustandsdarstellung. Wir zeigen empirisch, dass die Anwendung

von KF-Schichten inbesondere bei solchen Aufgabe effektiv ist, bei denen der mit Unsicherheit für die Entscheidungsfindung entscheidend ist.

Acknowledgements

I would like to thank my academic advisor Prof. Jan Peters for his guidance and support. I learned to value his vision and pragmatism for research. At the same time, under Jan's supervision I always felt empowered to pursue my own research ideas. I would also like to thank Professors Marc Bellemare, Kristian Kersting and Carsten Binnig for reviewing my thesis and serving as committee members.

I was fortunate to receive supervision from Julia Vinogradska and Felix Berkenkamp from the Bosch Center for AI. Their continued support was invaluable to me and their teachings have shaped me as a scientist. Julia taught me perseverance, thoroughness, clear writing and to strive for excellence in research. From Felix I learned how to effectively manage my projects, from asking myself the right questions to conducting experiments at scale. Beyond that, Julia and Felix genuinely cared for my personal development and well-being.

I am deeply grateful for starting my PhD alongside Alessandro Bottero. Alex was always open to discuss research ideas and provide thoughtful feedback. I learned from him many good practices for writing mathematical proofs. Beyond research, Alex was caring and supportive during rough patches in my PhD journey.

While doing my PhD I had the pleasure of interacting with outstanding people. Special thanks to Michael Volpp, Jan Woehlke, Felix Richter, Ilya Kamenshchikov and Christian Schorr from the Bosch RL team. I appreciate all the discussions and dinners we shared over the years. Many thanks as well to my group leader, Luigi Palmieri, for his support and encouragement in the last stages of my PhD. I would also like to thank the members of the IAS group from TU Darmstadt: thank you for being welcoming during the annual retreats. Special thanks as well to Nanette Schreiber, Sabine Schnitt and Heike Schmitt-Spall for helping me navigate all the administrative processes in TU Darmstadt.

To Brandon Sanderson, whose fantasy novels accompanied me for a good chunk of my PhD years and taught me valuable lessons. Life before death, strength before weakness, journey before destination.

To my parents, Ángela and Henry. From an early age you taught me to work hard and follow my dreams. Sadly, pursuing my passions drove me physically away from you but it was necessary to get closer to the man I want to be. I am forever indebt for all the oportunities you gave me in life and your unconditional love. To my sister, Adriana, thank you for being a role model and a source of wisdom. To the rest of my family and friends, thanks for all your support despite the distance.

To my girlfriend, María del Mar. Thank you for believing in me when I had a hard time believing in myself. Your unwavering love was a shining light in my darkest moments. All the happy moments alongside you made my PhD a more enjoyable experience. And last but not least, I am grateful you brought our lovely cat Arya into my life. Watching her sleep brings me peace and joy after a hard day of work.

Contents

List of Symbols

The following list introduces the common notation used throughout this thesis. Due to the large amount of variables, some symbols are overloaded across chapters, in which case the correct meaning will be apparent from the context of the respective chapter.

Symbol	Description
x	Scalar
X	Random variable
$X \sim \nu$	X is distributed according to distribution ν
$\mathcal{X}$	Set
$\mathbf{x}$	Vector
$\mathbf{X}$	Matrix
$\mathbb{P}(X)$	Probability density function of X
$\mathbb{E}_X[f(X)]$	Expected value of $f(X)$
$\mathbb{V}_X[f(X)]$	Variance of $f(X)$
$\mathbb{E}_{x\sim\nu}[f(x)]$	Expected value of $f(x)$ where x is sampled from ν
$\mathbb{V}_{x\sim\nu}[f(x)]$	Variance of $f(x)$ where x is sampled from ν
$f_{\#}\nu$	Pushforward distribution of ν by the function f
$\mathbb{R}$	Set of real numbers
$\mathcal{P}(\mathcal{X})$	Space of probability distributions over the set $\mathcal{X}$
$\mathcal{M}$	Markov Decision Process (MDP) or Partially Observable MDP (POMDP)
$\mathcal{A}$	MDP's action space
$\mathcal{S}$	MDP's state space
$\mathcal{O}$	POMDP's observation space
γ	MDP's discount factor
$r(\cdot,\cdot)$	MDP's reward function
$\rho(\cdot)$	MDP's Initial state distribution

$p(\cdot \mid \cdot, \cdot)$	MDP's transition function
$P(\cdot \mid \cdot, \cdot)$	MDP's random transition function
$\Phi(P)$	Prior distribution over transition function
$\mathcal{D}$	Dataset of collected MDP transitions
$\Phi(P \mid \mathcal{D})$	Posterior distribution over transition function conditioned on dataset $\mathcal{D}$
$\mathscr{P}$	Set of all transition functions
π	Stochastic control policy
$v^{\pi,p}$	Scalar value function of policy π under transition function p
Z^{π}	Random return function of policy π
V^{π}	Random value function of policy π under random transition function P
μ^{π}	Value distribution of policy π under random transition function P
$\overset{D}{=}$	Equality in distribution
$\mathcal{T}$	Bellman Operator

1. Introduction

Over the past decades, there has been a wide range of real-world applications of autonomous agents. From car manufacturing to warehouse management, current autonomous agents can execute high-precision tasks with super-human speed and reliability. Oftentimes, such applications are possible thanks to precise mathematical models of the system's dynamics and predictable surroundings. However, as we move to more complex applications, such precise models become prohibitive and classical approaches less effective. A more general solution requires agents that *efficiently learn* and *adapt* in *unknown* environments.

Reinforcement learning (RL) (Sutton and Barto, 2018) has gained popularity as a paradigm to train agents for optimal decision-making through interactions with *a priori* unknown environments. Despite many successful applications of RL in simulated environments like video-games (Mnih et al., 2013) and board-games (Tesauro, 1995), applying RL algorithms more broadly faces many challenges, from which we highlight the following:

Challenge #1: Sample-efficiency. In problems where collecting data is expensive, e.g., due to slow simulators or having to interact with a physical system, RL algorithms are required to be *sample-efficient*. That is, a performant policy must be learned with the fewest environment interactions possible. Many sub-challenges stem from sample-efficient learning, such as efficient exploration and leveraging historical data (also known as offline RL).

Challenge #2: Partial observability. When the environment observations only convey limited information for making optimal decisions, then RL algorithms must deal with *partial observability*. In such settings, the RL agent must *infer* the state of the system from past observations and act based on it. Partial observability is common in many practical scenarios, e.g., due to having access to limited sensors or because the dynamics of the system change over time or across evaluation episodes.

The central argument of this dissertation is that proper handling of *uncertainty* offers a common framework to tackle the aforementioned challenges. Our main research question is the following:

How can we efficiently estimate and leverage uncertainty in reinforcement learning algorithms?

To address this question, in this thesis we develop, evaluate and analyze uncertainty quantification techniques that consider the sequential nature of decision-making, such that the estimated uncertainty can be leveraged to improve the learning process.

We identify two types of uncertainty in RL: *aleatoric* uncertainty, which models the inherent noise of the environment, and *epistemic* uncertainty, which captures the agent's lack of knowledge about the dynamics (Kiureghian and Ditlevsen, 2009). While aleatoric uncertainty is irreducible, epistemic uncertainty can be diminished by strategically collecting more data. Thus, efficient learning algorithms isolate epistemic uncertainty and leverage it for informed decision-making (Gal, 2016).

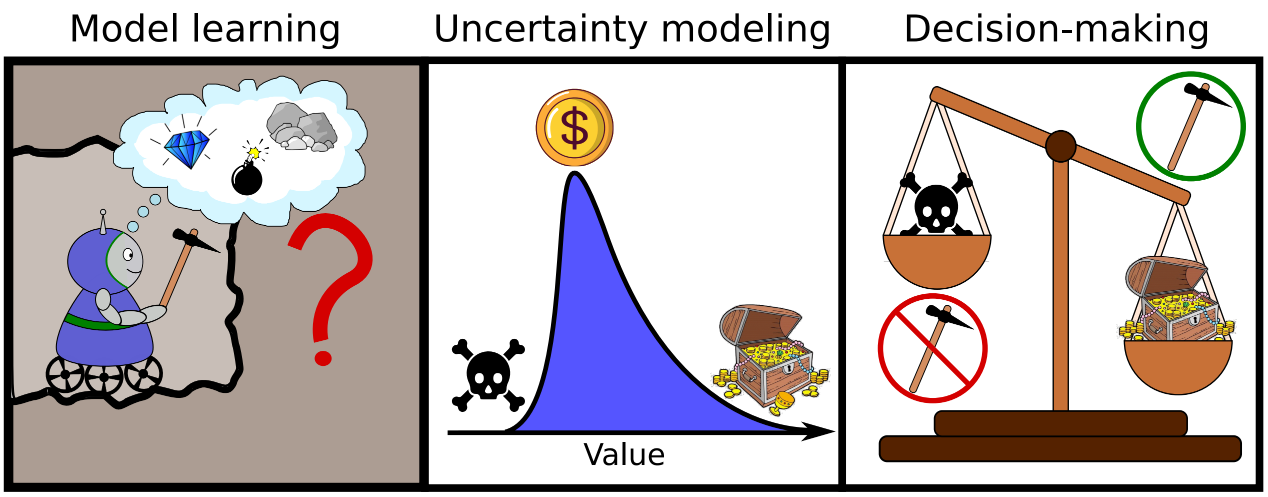


Figure 1.1.: We define uncertainty-aware decision-making in three iterative steps. **(Left)** Model learning uses prior knowledge and observed data to learn a probabilistic model of the environment. **(Middle)** Uncertainty from the learned model is propagated through the decision process to reason about the *long-term* value of a given behavior. **(Right)** The control policy is updated considering the value uncertainty.

In the first part of this thesis (Chapters 2 and 3) we assume full observability and consider the problem of estimating epistemic uncertainty around the long-term *value* of a control policy and how to leverage those estimates for sample-efficient RL. We adopt the Model-Based RL (MBRL) framework that learns a probabilistic model of the environment dynamics from collected data and naturally captures both sources of uncertainty (Sutton, 1991; Strehl and Littman, 2008). Current MBRL algorithms can solve high-dimensional tasks with unparalleled sample-efficiency by disentangling the aleatoric and epistemic uncertainties when training a control policy (Chua et al., 2018; Curi et al., 2020). One core challenge in MBRL is propagation of model uncertainty for *long-term reasoning* (Deisenroth and Rasmussen, 2011). We develop uncertainty quantification methods that act as a bridge between learning dynamic models and optimizing a policy. That is, our methods translate the uncertainty of the learned model into quantities that can be readily utilized for training a control policy. In Figure 1.1, we depict a general framework for the uncertainty-aware MBRL algorithms introduced in this thesis. To illustrate the approach, we use a running example of an autonomous robot tasked with exploring a mine in search for treasures.

Model learning. From prior knowledge and observed data, the purpose of model learning is to build a probabilistic model of the environment. Importantly, the learned model must capture the epistemic uncertainty due to limited data, such as not knowing what lies beyond a rock wall. Calibrated uncertainties in the learned model are key to enable the rest of the pipeline. While there exists lots of open questions regarding modelling of uncertainty in learned dynamics, this topic is not a core focus in this thesis. Instead, we start from standard practices for model learning and focus on how to leverage this knowledge for uncertainty representation and policy optimization.

Uncertainty Quantification. This module enables reasoning about the long-term *value* of different behaviors. In our example, if the robot continues digging through the mine, how likely is it that it will hit a bomb and explode? Or how likely is it that it will find diamonds and other treasures? A proper answer to these questions requires propagating the model uncertainty through the sequential decision process. Moreover, such uncertainty propagation needs to be efficient, scalable and easily integrated with current MBRL methodologies.

Decision-making. The last step of the pipeline is to consider the estimated values and their uncertainty to make a decision. The updated control policy depends entirely on the inherent objective of the agent. In our running example, as shown in the right diagram of Figure 1.1, the agent has a risk-seeking objective such that it values more the high potential of finding treasures than the low probability of death by detonating a bomb; as such, its final decision is to continue digging through the mine.

While we initially assume full observability, in the second part of the thesis (Chapter 4) we study the role of uncertainty in model-free RL under partial observability. Coming back to our running example of the autonomous robot, partial observability may exist due to a variety of reasons including observation noise, limited sensors and changing dynamics across deployments, just to name a few. For example, if the sensors are limited and/or noisy, the robot needs to aggregate data over multiple steps and across multiple sensors to produce a state representation amenable for navigating the mine. In this setting, we study uncertainty quantification of the inferred latent state of the system.

1.1. Contributions

In this dissertation, we investigate uncertainty estimation methods tailored for sequential decision-making tasks. Below is the summary of contributions of this thesis.

Chapter 2: We estimate the *variance* around the predicted performance (value) of a policy. The resulting uncertainty quantification method recovers the exact epistemic variance of the policy's value under certain assumptions. The predicted variance is readily integrated into RL algorithms for uncertainty-aware policy optimization. Furthermore, our method can be scaled using function approximation to tackle complex continuous-control problems. Evaluation in both tabular and continuous problems demonstrate improved sample-efficiency in difficult exploration problems. Moreover, we can use the same uncertainty-aware objectives for improved performance in offline optimization. This chapter is based on two articles (Luis et al., 2023a, 2024a), the former published at the *International Conference on Artificial Intelligence and Statistics* (AISTATS) and the latter submitted to Springer *Machine Learning Journal* (MLJ).

Chapter 3: We show how to efficiently learn the entire *distribution* of policy values, induced by the epistemic uncertainty of the learned dynamics model. The resulting method predicts a richer representation of uncertainty than the one developed in Chapter 2 and can accomodate more diverse policy optimization objectives. Extensive empirical evaluation shows improved final performance and sample efficiency in diverse continuous control tasks. The content of this chapter is based on the pre-print (Luis et al., 2023b) accepted to the *Journal of Machine Learning Research* (JMLR).

Chapter 4: We introduce a simple recurrent layer based on Kalman filtering that performs closed-form probabilistic inference in a latent state-space model. Our proposed Kalman filter (KF) layer can be easily integrated in existing model-free deep RL architectures where it is trained end-to-end for return maximization. The KF layer outputs a *filtered* state representation which is then used by a downstream RL algorithm. Our method excels in various problems where probabilistic reasoning is key for decision-making. This chapter is based on the pre-print (Luis et al., 2024b), submitted to the *Transactions of Machine Learning Research* (TMLR).

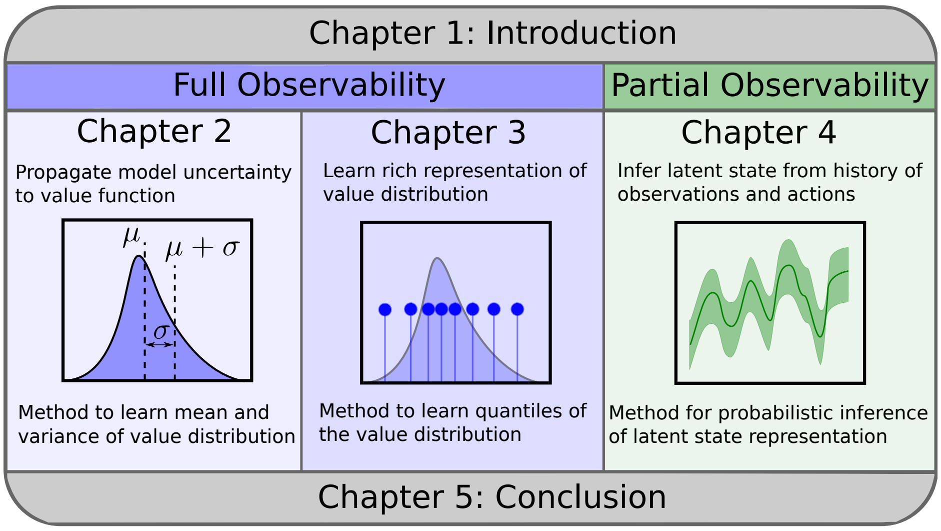

Chapter 1: Introduction
Full Observability
Partial Observability
Chapter 2
Propagate model uncertainty to value function
μ
$\mu + \sigma$
σ
Method to learn mean and variance of value distribution
Chapter 3
Learn rich representation of value distribution
Method to learn quantiles of the value distribution
Chapter 4
Infer latent state from history of observations and actions
Method for probabilistic inference of latent state representation
Chapter 5: Conclusion

Figure 1.2.: Outline of thesis. Chapters 2 and 3 consider uncertainty modelling in model-based RL under full observability, while Chapter 4 studies uncertainty representations in inferred latent states for model-free RL under partial observability.

1.2. Outline

The outline of the thesis is illustrated in Figure 1.2. The thesis is subdivided into two parts according to the type of RL problem being considered. The first part includes Chapters 2 and 3, where the core topic of study is uncertainty modelling in RL with full observability. That is, we observe the state of the system directly and the dynamics are fixed across evaluation episodes. In this context, both chapters consider the problem of modelling epistemic uncertainty around the long-term performance predictions of the RL agent. In Chapter 2, the proposed solution is to only model the mean and variance of the value function distribution, while Chapter 3 considers a richer representation of distributions.

In the second part of the thesis, composed by Chapter 4, we consider RL under partial observability, where the agent must learn a state representation from past observations and use this information for decision-making.

Lastly, in Chapter 5, we summarize the main findings of this thesis, discuss open problems and suggest avenues for future work.

2. Model-Based Epistemic Variance of Values for Risk-Aware Policy Optimization

We consider the problem of quantifying uncertainty over expected cumulative rewards in model-based reinforcement learning. In particular, we focus on characterizing the *variance* over values induced by a distribution over MDPs. Previous work upper bounds the posterior variance over values by solving a so-called uncertainty Bellman equation (UBE), but the over-approximation may result in inefficient exploration. We propose a new UBE whose solution converges to the true posterior variance over values and leads to lower regret in tabular exploration problems. We identify challenges to apply the UBE theory beyond tabular problems and propose a suitable approximation. Based on this approximation, we introduce a general-purpose policy optimization algorithm, Q-Uncertainty Soft Actor-Critic (QU-SAC), that can be applied for either risk-seeking or risk-averse policy optimization with minimal changes. Experiments in both online and offline RL demonstrate improved performance compared to other uncertainty estimation methods.

2.1. Introduction

The goal of reinforcement learning (RL) is optimal decision-making in an *a priori* unknown Markov Decision Process (MDP) (Sutton and Barto, 2018). The RL agent obtains rewards from interactions with the MDP and optimality is defined by some utility function of the accumulated rewards (also known as return). The return is, in general, a random variable due to two distinct types of uncertainty: *aleatoric*, induced by a combination of the agent's stochastic action selection and the random MDP state transitions; and *epistemic*, due to having limited data of the unknown MDP (Kiureghian and Ditlevsen, 2009). Aleatoric uncertainty is irreducible, since it is an inherent property of the problem, while epistemic uncertainty can be reduced by further interactions with the MDP. A standard utility in the RL literature is the *expected* return, known as the value function, which is a *risk-neutral* objective that averages over the *aleatoric* uncertainty without explicit treatment of epistemic uncertainty. In this chapter, we first present a method to explicitly estimate epistemic uncertainty around the value function and then argue for *epistemic risk-aware objectives* as a unified framework for tackling problems in which risk-neutrality leads to sub-optimal solutions.

We motivate the need for risk-aware objectives with two concrete practical tasks: online exploration and offline optimization. In online exploration, the MDP's reward signal is sparse and standard RL algorithms based on maximizing expected return converge to a suboptimal solution even in simple tasks (Raffin et al., 2021). In offline optimization, the RL agent does not interact with the MDP and solely relies on a dataset with limited support; in this case, standard RL algorithms without additional regularization are known to diverge (Levine et al., 2020). While each of these problems have been tackled individually in the past, we propose a unified solution by quantifying epistemic uncertainty and optimizing a simple risk-aware objective: risk-seeking to encourage exploration in the absence of a reward signal, or risk-averse for explicit regularization in offline optimization. The two behaviors are controlled by a single hyperparameter, thus the same algorithm can be applied to both exploration and offline problems.

In order to model epistemic uncertainty, we adopt the model-based RL (MBRL) paradigm, in which the RL agent learns a probabilistic model of the MDP (Sutton, 1991). For tabular RL problems with finite state-action spaces, provably efficient RL algorithms leverage the learned model of the MDP to derive epistemic-uncertainty-based rewards that instill exploratory behaviour (Strehl and Littman, 2008; Jaksch et al., 2010). Beyond these tabular RL approaches, modern deep learning MBRL methods quantify epistemic and aleatoric uncertainty in the learned MDP dynamics (Depeweg et al., 2018; Chua et al., 2018) and leverage them to optimize the policy (Curi et al., 2020). Still, proper uncertainty quantification of long-term return predictions remains a challenging trade-off between accuracy and tractable probabilistic inference (Deisenroth and Rasmussen, 2011). Despite these challenges, it has been shown that quantification of uncertainty around the policy's value enables risk-awareness, i.e., reasoning about the long-term risk of rolling out a policy. Promising results have been reported for both risk-seeking (Deisenroth and Rasmussen, 2011; Fan and Ming, 2021) and risk-averse (Zhou et al., 2020; Yu et al., 2020) policy optimization.

Similar to prior work in MBRL, we use a Bayesian approach to characterize uncertainty in the MDP via a posterior distribution (Dearden et al., 1999). This distributional perspective of the RL environment induces distributions over functions of interest for solving the RL problem, e.g., the value function. Our perspective differs from *distributional* RL (Bellemare et al., 2017), whose main object of study is the *aleatoric* noise around the *return*. As such, distributional RL models *aleatoric* uncertainty, whereas our Bayesian MBRL perspective focuses on the *epistemic* uncertainty arising from finite data of the underlying MDP.

In this chapter, we analyze the *variance* of the distribution over value functions and design an algorithm to estimate it. Our method relies on dynamic programming and the well-known Bellman equation (Bellman, 1957). In particular, previous work by O'Donoghue et al. (2018); Zhou et al. (2020) showed that the dynamic programming solution to a so-called uncertainty Bellman equation (UBE) is a guaranteed upper-bound on the posterior variance of the value function. Our theoretical result is a new UBE whose solution is *exactly* the posterior variance of the value function, thus closing the theoretical gap in previous work. Beyond the theoretical analysis, we further identify limitations of naive applications of our theoretical result with neural networks as function approximators and propose a novel solution. Our aim is to devise a general algorithm for RL problems where optimizing for risk-neutral objectives is known to underperform. In particular, we consider two such problems: exploration, which typically requires risk-seeking behaviour, and offline optimization, which benefits from risk-averse objectives.

Our contribution. Based on the UBE framework, we first present a result that closes the theoretical gap in previous upper-bounds. Our epistemic variance estimate shows improved regret when used as a signal for exploration in tabular problems. Second, we identify challenges in applying the UBE theory to practical problems and propose suitable approximations. The result is a general-purpose algorithm called Q-Uncertainty Soft Actor-Critic (QU-SAC) that can be applied for either risk-seeking or risk-averse policy optimization with minimal changes. We evaluate QU-SAC in exploration tasks from the DeepMind Control (DMC) suite (Tunyasuvunakool et al., 2020) as well as offline RL tasks from the D4RL benchmark (Fu et al., 2020).

2.1.1. Related work

Model-free Bayesian RL. Model-free approaches to Bayesian RL directly model the distribution over values, e.g., with normal-gamma priors (Dearden et al., 1998), Gaussian Processes (Engel et al., 2003) or ensembles of neural networks (Osband et al., 2016). Jorge et al. (2020) estimate value distributions using a backwards induction framework, while Metelli et al. (2019) propagate uncertainty using Wasserstein barycenters. Fellows et al. (2021) showed that, due to bootstrapping, model-free Bayesian methods infer a posterior over Bellman operators rather than values.

Model-based Bayesian RL. Model-based Bayesian RL maintains a posterior over plausible MDPs given the available data, which induces a distribution over values. The MDP uncertainty is typically represented in the one-step transition model as a by-product of model-learning. For instance, the well-known PILCO algorithm by Deisenroth and Rasmussen (2011) learns a Gaussian Process (GP) model of the transition dynamics and integrates over the model's total uncertainty to obtain the expected values. In order to scale to high-dimensional continuous control problems, Chua et al. (2018) propose PETS, which uses ensembles of probabilistic neural networks (NNs) to capture both aleatoric and epistemic uncertainty as first proposed by Lakshminarayanan et al. (2017). Both approaches propagate model uncertainty during policy evaluation and improve the policy via greedy exploitation over this model-generated noise. Dyna-style (Sutton, 1991) actor-critic algorithms have been paired with model-based uncertainty estimates for improved performance in both online (Buckman et al., 2018; Zhou et al., 2019) and offline (Yu et al., 2020; Kidambi et al., 2020) RL.

Online RL - Optimism. To balance exploration and exploitation, provably-efficient RL algorithms based on *optimism in the face of the uncertainty* (OFU) (Auer and Ortner, 2006; Jaksch et al., 2010) rely on building upper-confidence (optimistic) estimates of the true values. These optimistic values correspond to a modified MDP where the rewards are enlarged by an uncertainty bonus, which encourages exploration. In practice, however, the aggregation of optimistic rewards may severely over-estimate the true values, rendering the approach inefficient (Osband and Van Roy, 2017). O'Donoghue et al. (2018) show that methods that approximate the variance of the values can result in much tighter upper-confidence bounds, while Ciosek et al. (2019) demonstrate their use in complex continuous control problems. Similarly, Chen et al. (2017) propose a model-free ensemble-based approach to estimate the variance of values.

Offline RL - Pessimism. In offline RL, the policy is optimized solely from offline (static) data rather than from online interactions with the environment (Levine et al., 2020). A primary challenge in this setting is known as *distribution shift*, which refers to the shift between the state-action distribution of the offline dataset and that of the learned policy. The main underlying issue with distribution shifts in offline RL relates to querying value functions out-of-distribution (OOD) with no opportunity to correct for generalization errors via online interactions (as in the typical RL setting). One prominent technique to deal with distribution shifts is known as *conservatism* or *pessimism*, where a pessimistic value function (typically a lower bound of the true values) is learned by regularizing OOD actions (Kumar et al., 2020; Bai et al., 2022). Model-based approaches to pessimism can be sub-divided into uncertainty-free (Yu et al., 2021; Rigter et al., 2022) and uncertainty-based methods (Yu et al., 2020; Kidambi et al., 2020; Jeong et al., 2023). While uncertainty-free pessimism circumvents the need to explicitly estimate the uncertainty, the current state-of-the-art method CBOP (Jeong et al., 2023) is uncertainty-based. Our QU-SAC algorithm falls into the uncertainty-based category and differentiates from prior work over which uncertainty it estimates: MOPO (Yu et al., 2020) uses the maximum aleatoric standard deviation of a dynamics ensemble forward prediction, MOREL (Kidambi et al., 2020) is similar but uses the maximum pairwise difference of the mean predictions, CBOP (Jeong et al., 2023) instead does approximate Bayesian inference directly on the Q-value predictions conditioned on empirical (bootstrapped) return estimates. Instead, QU-SAC learns a Bayesian estimate of the Q-values variance via approximately solving a UBE. To the best of our knowledge, this is the first time a UBE-based algorithm is used for offline RL.

Unified offline / online RL. The closest body of work in which offline and online optimization are treated under the same umbrella is that of offline-to-online RL, also known as online fine-tuning (Lee et al., 2022; Nakamoto et al., 2023). Lei et al. (2024) unify the offline and online phases under the same objective function, but the training procedure between both phases differs, adding further complexity. Zhao et al. (2023) use the same base algorithm (SAC) in both phases, but risk-awareness is procured by different methods: CQL (Kumar et al., 2020) in the offline phase, and SUNRISE (Lee et al., 2021) for online fine-tuning.

Uncertainty in RL. Interest about the higher moments of the *return* of a policy dates back to the work of Sobel (1982), showing these quantities obey a Bellman equation. Methods that leverage these statistics of the return are known as *distributional* RL (Tamar et al., 2013; Bellemare et al., 2017). Instead, we focus specifically on estimating and using the *variance* of the *expected return* for policy optimization. A key difference between the two perspectives is the type of uncertainty they model: distributional RL models the *aleatoric* uncertainty about the returns, which originates from the aleatoric noise of the MDP transitions and the stochastic policy; our perspective studies the *epistemic* uncertainty about the value function, due to incomplete knowledge of the MDP. Provably efficient RL algorithms use this isolated epistemic uncertainty as a signal to balance exploring the environment and exploiting the current knowledge.

UBE-based RL. O'Donoghue et al. (2018) propose a UBE whose fixed-point solution converges to a guaranteed upper-bound on the posterior variance of the value function in the tabular RL setting. This approach was implemented in a model-free fashion using the DQN (Mnih et al., 2013) architecture and showed performance improvements in Atari games. Follow-up work by Markou and Rasmussen (2019) empirically shows that the upper-bound is loose and the resulting over-approximation of the variance impacts negatively the regret in tabular exploration problems. Zhou et al. (2020) propose a modified UBE with a tighter upper-bound on the value function, which is then paired with proximal policy optimization (PPO) (Schulman et al., 2017) in a conservative on-policy model-based approach to solve continous-control tasks. Our QU-SAC algorithm integrates UBE-based uncertainty quantification into a model-based soft actor-critic (SAC) (Haarnoja et al., 2018) architecture similar to Janner et al. (2019); Froehlich et al. (2022).

2.2. Problem Statement

We consider an agent that acts in an infinite-horizon MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, p, \rho, r, \gamma\}$ with finite state space $\mathcal{S}$, finite action space $\mathcal{A}$, unknown transition function $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ that maps states and actions to the set of probability distributions over $\mathcal{S}$, an initial state distribution $\rho : \mathcal{S} \to [0, 1]$, a known and bounded reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and a discount factor $\gamma \in [0, 1)$. Although we consider a known reward function, the main theoretical results can be easily extended to the case where it is learned alongside the transition function (see Appendix A.2.1). The agent is equipped with an action-selection stochastic policy $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ that defines the conditional probability distribution $\pi(a \mid s)$, $(s, a) \in \mathcal{S} \times \mathcal{A}$. Given an initial state $S_0 \sim \rho$ and a policy π, the RL agent interacts with the environment and generates a random *trajectory* $T = \{S_h, A_h, R_h\}_{h=0}^{\infty}$, with $A_h \sim \pi(\cdot \mid S_h)$, $R_h = r(S_h, A_h)$, $S_{h+1} \sim p(\cdot \mid S_h, A_h)$ for $h \geq 0$. We define the value function $v^{\pi,p} : \mathcal{S} \to \mathbb{R}$ of a policy π and transition function p as the expected sum of discounted rewards under the MDP dynamics,

$$v^{\pi,p}(s) = \mathbb{E}_T\left[\sum_{h=0}^{\infty}\gamma^h R_h \middle| S_0 = s, p\right], \tag{2.1}$$

where the expectation is taken under the random trajectories T drawn from the trajectory distribution $\mathbb{P}(T) = \prod_{h=0}^{\infty}\pi(a_h \mid s_h)p(s_{h+1} \mid s_h, a_h)$.

2.2.1. Bayesian RL

We adopt a Bayesian perspective and model the unknown dynamics p as a random transition function P with some prior distribution $\Phi(P)$. As the agent acts in $\mathcal{M}$, it collects data[1] $\mathcal{D}$ and obtains the posterior distribution $\Phi(P \mid \mathcal{D})$ via Bayes' rule. More concretely, for tabular problems we consider priors that admit analytical posterior updates (e.g., Dirichlet, Gaussian) (Dearden et al., 1999), and for continuous state-action spaces we use neural network ensembles (Lakshminarayanan et al., 2017) which have been linked to approximate Bayesian inference (Osband et al., 2018).

In what follows, we will assume $P \sim \Phi(P \mid \mathcal{D})$ and consider trajectories T defined as previously but with next-states as $S_{h+1} \sim P(\cdot \mid S_h, A_h)$. Notably, the sampling process of next states mixes two sources of uncertainty: the aleatoric noise, as with the original MDP, but also the uncertainty in P due to finite data, often called *epistemic* uncertainty. Consequently, the aleatoric and epistemic noise in trajectories propagates to the returns. We define the value function of policy π as a random variable under the random dynamics P as

$$V^{\pi}(s) = v^{\pi,P}(s). \tag{2.2}$$

According to the value function definition in (2.1), V^{π} is an expectation over the trajectories T *conditioned* on the random variable P, which means the aleatoric noise of trajectories is averaged out, but the epistemic uncertainty (due to the conditioning on P) remains and is propagated to V^{π}. Intuitively, to obtain a sample from V^{π} is equivalent to sample from the posterior $\Phi(P \mid \mathcal{D})$ and calculate the corresponding expected return, i.e., the value. As such, the stochasticity of V^{π} vanishes as we gather data and $\Phi(P \mid \mathcal{D})$ concentrates around the true transition function p.

The main focus of this chapter is to study methods that estimate the *variance* of the random value function V^{π}, denoted $\mathbb{V}_P\left[V^{\pi}(s)\right]$. Our theoretical results extend to state-action value functions (see Appendix A.2.2). The motivation behind studying this quantity is its potential for risk-aware optimization.

A method to estimate an upper-bound of the variance of Q-values by solving a UBE was introduced by Zhou et al. (2020). Their theory holds for a class of MDPs where the value functions and transition functions are conditionally independent. This family of MDPs is characterized by the following assumptions:

Assumption 1 (Parameter Independence (Dearden et al., 1999)). *The posterior over the random vector $P(\cdot \mid s, a)$ is independent for each pair $(s, a) \in \mathcal{S} \times \mathcal{A}$.*

Assumption 2 (Directed Acyclic MDP (O'Donoghue et al., 2018)). *Let $\tilde{p} \in \mathscr{P}$ be a realization of the random variable P. Then, the MDP $\tilde{\mathcal{M}}$ with transition function $\tilde{p}$ is a directed acyclic graph, i.e., states are not visited more than once in any finite trajectory.*

[1]We omit time-step subscripts and refer to dataset $\mathcal{D}$ as the collection of all available transition data.

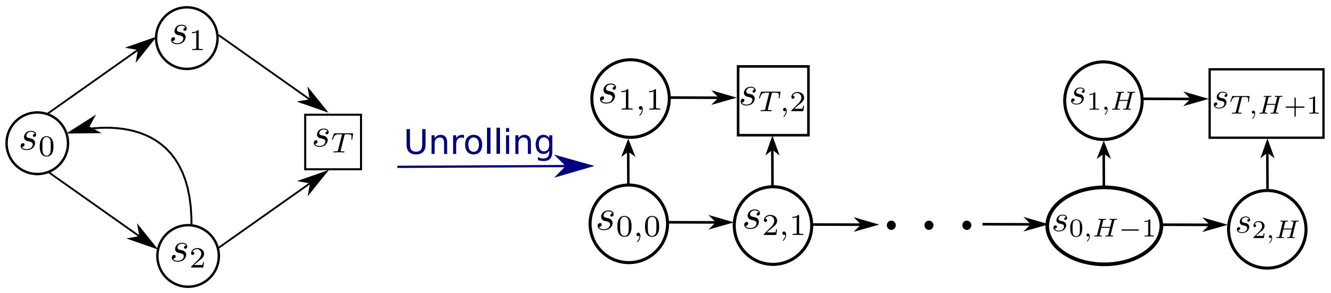


Figure 2.1.: Procedure of unrolling an MDP with cycles. We denote by $s_{i,k}$ the unrolled state which represents being in state s_i of the original MDP at time step k. The unrolled MDP is only an *approximation* of the original version due to truncation after finite H steps.

Assumption 3 (Terminal State). *Define a terminal (absorbing) state s_T such that $r(s_T, a) = 0$ and $p(s_T \mid s_T, a) = 1$ for any $a \in \mathcal{A}$ and $p \in \mathscr{P}$. Let $\tilde{p} \in \mathscr{P}$ be a realization of the random variable P. Then, the MDP $\tilde{\mathcal{M}}$ with transition function $\tilde{p}$ deterministically transitions to s_T after a finite horizon $H \leq |\mathcal{S}|$, i.e., $\tilde{p}(s_T \mid s_H, a) = 1$ for any $a \in \mathcal{A}$.*

The consequence of these assumptions is that $P(s' \mid s, a)$ and $V^\pi(s')$ are conditionally independent for all triplets (s, a, s') (see Lemma 3). Assumption 1 is satisfied when modelling state transitions as independent categorical random variables for every pair (s, a), with the unknown parameter vector $P(\cdot \mid s, a)$ under a Dirichlet prior (Dearden et al., 1999). Assumptions 2 and 3 imply that any infinite trajectory is composed of *distinct* states for the first H steps and then remains at the terminal state s_T indefinitely. Our theoretical results do not hold in the general case of MDPs with cycles. However, one may still obtain reasonable approximations by considering an equivalent time-inhomogeneous MDP without cycles (also known as the "unrolled" MDP (O'Donoghue et al., 2018)) by forcing a transition to a terminal state after H steps. In Figure 2.1 we show the unrolling procedure of an MDP that contains cycles. The two MDPs are *not* equivalent, but the unrolled approximation improves as $H \to \infty$, but in the limit implies an infinite state space, which would then require additional measure-theoretic considerations outside the scope of this work (cf. Bellemare et al., 2023, Remark 2.3). While unrolling the MDP is rarely done in practice, it serves as a reasonable approximation to extend our theoretical results to the general setting of MDPs that contain cycles.

Other quantities of interest are the posterior mean transition function starting from the current state-action pair (s, a),

$$\bar{p}(\cdot \mid s, a) = \mathbb{E}_P\big[P(\cdot \mid s, a)\big], \tag{2.3}$$

and the posterior mean value function for any $s \in \mathcal{S}$,

$$\bar{v}^\pi(s) = \mathbb{E}_P\big[V^\pi(s)\big]. \tag{2.4}$$

Note that $\bar{p}$ is a transition function that combines both aleatoric *and* epistemic uncertainty. Even if we limit the posterior Φ to only include deterministic transition functions, $\bar{p}$ remains a stochastic transition function due to the epistemic uncertainty.

In prior work by Zhou et al. (2020), *local* uncertainty is defined as

$$w(s) = \mathbb{V}_P\Big[\sum\nolimits_{a,s'} \pi(a \mid s) P(s' \mid s, a) \bar{v}^\pi(s')\Big], \tag{2.5}$$

which captures variability of the posterior mean value function at the next state s'. Based on this local uncertainty, Zhou et al. (2020) propose the UBE

$$W^{\pi}(s) = \gamma^2 w(s) + \gamma^2 \sum_{a,s'} \pi(a \mid s)\bar{p}(s' \mid s,a) W^{\pi}(s'), \tag{2.6}$$

that propagates the local uncertainty using the posterior mean dynamics. It was proven that the fixed-point solution of (2.6) is an upper-bound of the epistemic variance of the values, i.e., it satisfies $W^{\pi}(s) \geq \mathbb{V}_P\big[V^{\pi}(s)\big]$ for all s.

2.3. Uncertainty Bellman Equation

In this section, we build a new UBE whose fixed-point solution is *equal* to the variance of the value function and we show explicitly the gap between (2.6) and $\mathbb{V}_P\big[V^{\pi}(s)\big]$.

The values $v^{\pi,p}$ are the fixed-point solution to the Bellman expectation equation, which relates the value of the current state s with the value of the next state s'. Further, under Assumptions 1–3, applying the expectation operator to the Bellman recursion results in $\bar{v}^{\pi}(s) = v^{\pi,\bar{p}}(s)$. The Bellman recursion propagates knowledge about the *local* rewards $r(s,a)$ over multiple steps, so that the value function encodes the *long-term* value of states if we follow policy π. Similarly, a UBE is a recursive formula that propagates a notion of *local uncertainty*, $u(s)$, over multiple steps. The fixed-point solution to the UBE, which we call the U-values, encodes the *long-term epistemic uncertainty* about the values of a given state.

Previous formulations by O'Donoghue et al. (2018); Zhou et al. (2020) differ only on their definition of the local uncertainty and result on U-values that upper-bound the posterior variance of the values. The first key insight is that we can define u such that the U-values converge exactly to the variance of values. This result is summarized in the following theorem:

Theorem 1. *Under Assumptions 1–3, for any $s \in \mathcal{S}$ and policy π, the posterior variance of the value function, $U^{\pi} = \mathbb{V}_P[V^{\pi}]$ obeys the uncertainty Bellman equation*

$$U^{\pi}(s) = \gamma^2 u(s) + \gamma^2 \sum_{a,s'} \pi(a \mid s)\bar{p}(s' \mid s,a) U^{\pi}(s'), \tag{2.7}$$

where $u(s)$ is the local uncertainty defined as

$$u(s) = \mathbb{V}_{a,s'\sim\pi,\bar{p}}\big[\bar{v}^{\pi}(s')\big] - \mathbb{E}_P\Big[\mathbb{V}_{a,s'\sim\pi,P}\big[V^{\pi}(s')\big]\Big]. \tag{2.8}$$

Proof. See Appendix A.1.1. □

One may interpret the U-values from Theorem 1 as the associated state-values of an alternate *uncertainty MDP*, $\mathcal{U} = \big\{\mathcal{S}, \mathcal{A}, \bar{p}, \rho, \gamma^2 u, \gamma^2\big\}$, where the agent receives uncertainty rewards and transitions according to the mean dynamics $\bar{p}$.

A key difference between u and w is how they represent epistemic uncertainty: in the former, it appears only within the first term, through the one-step variance over $\bar{p}$; in the latter, the variance is computed over Φ. While the two perspectives may seem fundamentally different, in the following theorem we present a clear relationship that connects Theorem 1 with the upper bound (2.6).

Theorem 2. *Under Assumptions 1–3, for any $s \in \mathcal{S}$ and policy π, it holds that $u(s) = w(s) - g(s)$, where $g(s) = \mathbb{E}_P\Big[\mathbb{V}_{a,s' \sim \pi, P}\big[V^\pi(s')\big] - \mathbb{V}_{a,s' \sim \pi, P}\big[\bar{v}^\pi(s')\big]\Big]$. Furthermore, we have that the gap $g(s)$ is non-negative, thus $u(s) \leq w(s)$.*

Proof. See Appendix A.1.2. □

The gap $g(s)$ of Theorem 2 can be interpreted as the *average difference* of aleatoric uncertainty about the next values with respect to the mean values. The gap vanishes only if the epistemic uncertainty goes to zero, or if the MDP and policy are both deterministic.

We directly connect Theorems 1 and 2 via the equality

$$\underbrace{\mathbb{V}_{a,s' \sim \pi, \bar{p}}\big[\bar{v}^\pi(s')\big]}_{\text{total}} = \underbrace{w(s)}_{\text{epistemic}} + \underbrace{\mathbb{E}_P\Big[\mathbb{V}_{a,s' \sim \pi, P}\big[\bar{v}^\pi(s')\big]\Big]}_{\text{aleatoric}}, \tag{2.9}$$

which helps us analyze our theoretical results. The uncertainty reward defined in (2.8) has two components: the first term corresponds to the *total uncertainty* about the *mean* values of the next state, which is further decomposed in (2.9) into an epistemic and aleatoric components. When the epistemic uncertainty about the MDP vanishes, then $w(s) \to 0$ and only the aleatoric component remains. Similarly, when the MDP and policy are both deterministic, the aleatoric uncertainty vanishes and we have $\mathbb{V}_{a,s' \sim \pi, \bar{p}}\big[\bar{v}^\pi(s')\big] = w(s)$. The second term of (2.8) is the *average aleatoric uncertainty* about the value of the next state. When there is no epistemic uncertainty, this term is non-zero and exactly equal to the alectoric term in (2.9) which means that $u(s) \to 0$. Thus, we can interpret $u(s)$ as a *relative* local uncertainty that subtracts the average aleatoric noise out of the total uncertainty around the mean values. Perhaps surprisingly, our theory allows negative $u(s)$ (see Section 2.3.1 for a concrete example).

Through Theorem 2 we provide an alternative proof of why the UBE (2.6) results in an upper-bound of the variance, specified by the next corollary.

Corollary 1. *Under Assumptions 1–3, for any $s \in \mathcal{S}$ and policy π, it holds that the solution to the uncertainty Bellman equation* (2.6) *satisfies $W^\pi(s) \geq U^\pi(s)$.*

Proof. The solution to the Bellman equations (2.6) and (2.7) are the value functions under some policy π of identical MDPs except for their reward functions. Given two identical MDPs $\mathcal{M}_1$ and $\mathcal{M}_2$ differing only on their corresponding reward functions r_1 and r_2, if $r_1 \leq r_2$ for any input value, then for any trajectory τ we have that the returns (sum of discounted rewards) must obey $z_1(\tau) \leq z_2(\tau)$. Lastly, since the value functions v_1^π, v_2^π are defined as the expected returns under the same trajectory distribution and the expectation operator preserves inequalities, then we have that $z_1(\tau) \leq z_2(\tau) \implies v_1^\pi \leq v_2^\pi$. □

Corollary 1 reaches the same conclusions as Zhou et al. (2020), but it brings important explanations about their upper bound on the variance of the value function. First, by Theorem 2 the upper bound is a consequence of the over approximation of the reward function used to solve the UBE. Second, the gap between the exact reward function $u(s)$ and the approximation $w(s)$ is fully characterized by $g(s)$ and brings interesting insights. In particular, the influence of the gap term depends on the stochasticity of the dynamics and the policy. In the limit, the term vanishes under deterministic transitions and action selection. In this scenario, the upper-bound found by Zhou et al. (2020) becomes tight.

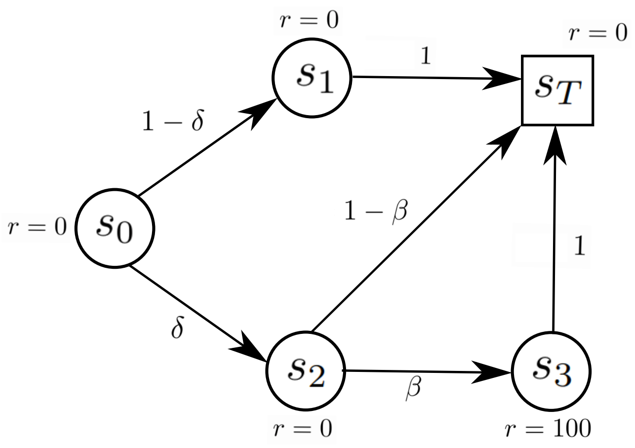


Figure 2.2.: Toy example Markov Reward Process. The random variables δ and β indicate epistemic uncertainty about the MRP's transition probabilities. State s_T is an absorbing (terminal) state.

Table 2.1.: Comparison of local uncertainty rewards and solutions to the UBE associated with the toy example from Figure 2.2. The U-values converge to the true posterior variance of the values, while W^π obtains an upper-bound.

States	$u(s)$	$w(s)$	$W^\pi(s)$	$U^\pi(s)$
s_0	-0.6	5.0	21.3	15.7
s_2	25.0	25.0	25.0	25.0

Our method returns the exact *epistemic* uncertainty about the values by considering the inherent aleatoric uncertainty of the MDP and the policy. In a practical RL setting, disentangling the two sources of uncertainty is key for effective exploration. We are interested in exploring regions of high epistemic uncertainty, where new knowledge can be obtained. If the variance estimate fuses both sources of uncertainty, then we may be guided to regions of high uncertainty but with little information to be gained.

2.3.1. Toy Example

To illustrate the theoretical findings of this chapter, consider the simple Markov reward process (MRP) of Figure 2.2. Assume δ and β to be random variables drawn from a discrete uniform distribution $\delta \sim \text{Unif}(\{0.7, 0.6\})$ and $\beta \sim \text{Unif}(\{0.5, 0.4\})$. As such, the distribution over possible MRPs is finite and composed of the four possible combinations of δ and β. Note that the example satisfies Assumptions 1 and 2. In Table 2.1 we include the results for the uncertainty rewards and solution to the respective UBEs (the results for s_1 and s_3 are trivially zero). For state s_2, the upper-bound W^π is tight and we have $W^\pi(s_2) = U^\pi(s_2)$. In this case, the gap vanishes not because of lack of stochasticity, but rather due to lack of epistemic uncertainty about the next-state values. Indeed, the values for s_3 and s are independent of δ and β, which results in the gap terms for s_2 cancelling out. For state s_0 the gap is non-zero and W^π overestimates the variance of the value by $\sim 36\%$. Our UBE formulation prescribes a *negative* reward to be propagated in order to obtain the correct posterior variance.

Algorithm 1 Model-based Q-variance estimation

```
1: Input: Posterior MDP $\Gamma$, policy $\pi$.
2: $\{p_i, r_i\}_{i=1}^N \leftarrow$ sample_mdp($\Gamma$)
3: $\bar{Q}^\pi$, $\{Q_i\}_{i=1}^N \leftarrow$solve_bellman$\left(\{p_i, r_i\}_{i=1}^N, \pi\right)$
4: $\hat{U}^\pi \leftarrow$ qvariance$\left(\{p_i, r_i, Q_i\}_{i=1}^N, \bar{Q}^\pi, \pi\right)$
```

2.4. Uncertainty-Aware Policy Optimization

In this section, we propose techniques to leverage uncertainty quantification of Q-values (also known as state-action values) for both online and offline RL problems. In what follows, we consider the general setting with unknown rewards and define Γ to be the posterior distribution over MDPs, from which we can sample both reward and transition functions. Define $\hat{U}^\pi$ to be an estimate of the posterior variance over Q-values for some policy π. Then, we consider algorithms that perform policy updates via the following upper (or lower) confidence bound (Auer and Ortner, 2006) type of optimization problem

$$\pi = \operatorname{argmax}_\pi \bar{Q}^\pi + \lambda\sqrt{\hat{U}^\pi}, \tag{2.10}$$

where $\bar{Q}^\pi$ is the posterior mean Q-value function and λ is a risk-awareness parameter. A positive λ corresponds to risk-seeking, optimistic exploration while negative λ denotes risk-averse, pessimistic anti-exploration.

Algorithm 1 describes our general framework to estimate $\bar{Q}^\pi$ and $\hat{U}^\pi$: we sample an ensemble of N MDPs from the current posterior Γ in Line 2 and use it to solve the Bellman expectation equation in Line 3, resulting in an ensemble of N corresponding Q functions and the posterior mean $\bar{Q}^\pi$. Lastly, $\hat{U}^\pi$ is estimated in Line 4 via a generic variance estimation method `qvariance`. In what follows, we provide concrete implementations of `qvariance` both in tabular and continuous problems.

2.4.1. Tabular Problems

For problems with tabular representations of the state-action space, we implement `qvariance` by directly solving the proposed UBE[2](2.7), which we denote `exact-ube`. For this purpose, we impose a Dirichlet prior on the transition function and a standard Normal prior for the rewards (O'Donoghue et al., 2019), which leads to closed-form posterior updates. After sampling N times from the MDP posterior (Line 2), we obtain the Q-functions (Line 3) in closed-form by solving the corresponding Bellman equation. The uncertainty rewards are estimated via sample-based approximations of the expectations/variances therein. Lastly, we solve (2.10) via policy iteration until convergence is achieved or until a maximum number of steps is reached.

Practical bound. The choice of a Dirichlet prior violates Assumption 2. A challenge arises in this practical setting: `exact-ube` may result in *negative* U-values, as a combination of (i) the assumptions not holding and (ii) the possibility of negative uncertainty rewards. While (i) cannot be easily resolved, we propose a practical upper-bound on the solution of (2.7) such that the resulting U-values are non-negative and hence

[2]For the UBE-based methods we use the equivalent equations for Q-functions, see Appendix A.2.3 for details.

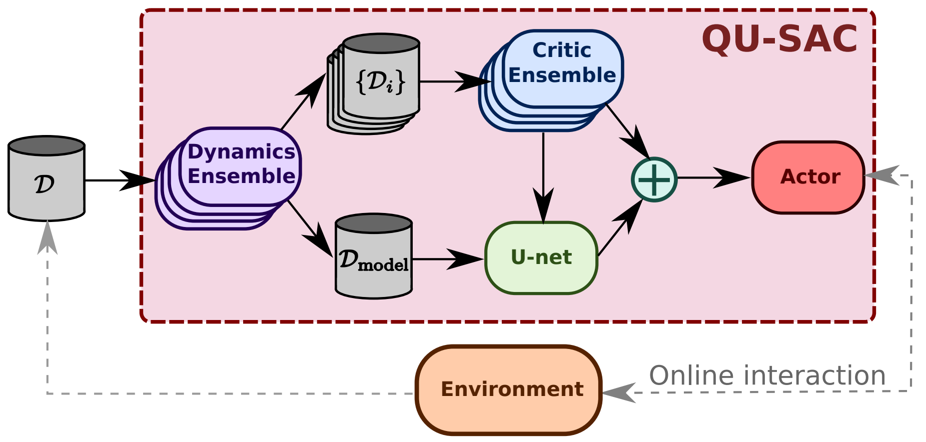


Figure 2.3.: Architecture for Q-Uncertainty Soft Actor-Critic (QU-SAC). The dataset $\mathcal{D}$ may be either static, as in offline RL, or be dynamically populated with online interactions. This dataset is used to train an ensemble of dynamics models which is then used for synthetic rollout generation. Each member of the ensemble populates its own buffer $\mathcal{D}_i$, which is used to train a corresponding ensemble of critics. Additionally, member-randomized rollouts are stored in $\mathcal{D}_{\text{model}}$ and used to train a U-net, which outputs an estimated epistemic variance of the value prediction. Lastly, the actor aims to optimize the risk-aware objective (2.10), which combines the output of the critic ensemble and the U-net.

interpretable as variance estimates. We consider the clipped uncertainty rewards $\tilde{u} = \max(u_{\text{min}}, u(s))$ with corresponding U-values $\tilde{U}^\pi$. It is straightforward to prove that, if $u_{\text{min}} = 0$, then $W^\pi(s) \geq \tilde{U}^\pi(s) \geq U^\pi(s)$, which means that using $\tilde{U}^\pi$ still results in a tighter upper-bound on the variance than W^π, while preventing non-positive solutions to the UBE. In what follows, we drop this notation and assume all U-values are computed from clipped uncertainty rewards.

2.4.2. Continuous Problems

We tackle problems in continuous domains using neural networks for function approximation. The resulting architecture is named Q-Uncertainty Soft Actor-Critic (QU-SAC) which builds upon MBPO by Janner et al. (2019) and is depicted in Figure 2.3.

Posterior dynamics. In contrast to the tabular implementation, maintaining an explicit distribution over MDPs from which we can sample is intractable. Instead, we approximate Γ with an ensemble, which have been linked to approximate posterior inference (Osband et al., 2018). More concretely, we model Γ as a discrete uniform distribution of N probabilistic neural networks, denoted p_θ, that output the mean and covariance of a Gaussian distribution over next states and rewards (Chua et al., 2018). In this case, the output of Line 2 in Algorithm 1 is precisely the ensemble of neural networks.

Critics. The original MBPO trains Q-functions represented as neural networks via TD-learning on data generated via *model-randomized* k-step rollouts from initial states that are sampled from $\mathcal{D}$. Each forward prediction of the rollout comes from a randomly selected model of the ensemble and the transitions are stored in a single replay buffer $\mathcal{D}_{\text{model}}$, which is then fed into a model-free optimizer like SAC. Algorithm 1 requires a few modifications from the MBPO methodology. To implement Line 3, in addition to $\mathcal{D}_{\text{model}}$, we create N new buffers $\{\mathcal{D}_i\}_{i=1}^N$ filled with *model-consistent* rollouts, where each k-step rollout is generated under a single model of the ensemble, starting from initial states sampled from $\mathcal{D}$. We train an ensemble of N value functions $\{Q_i\}_{i=1}^N$, parameterized by $\{\psi_i\}_{i=1}^N$, and minimize the residual Bellman error with entropy regularization

$$\mathcal{L}(\psi_i) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}_i}\left[\big(y_i - Q_i(s,a;\psi_i)\big)^2\right], \tag{2.11}$$

where $y_i = r + \gamma\big(Q_i(s', a'; \bar{\psi}_i) - \alpha \log \pi_\phi(a' \mid s')\big)$ and $\bar{\psi}_i$ are the target network parameters updated via Polyak averaging for stability during training (Mnih et al., 2013). The mean Q-values, $\bar{Q}^\pi$, are estimated as the average value of the Q-ensemble.

Uncertainty rewards. Our theory prescribes propagating the uncertainty rewards (2.8) to obtain the `exact-ube` estimate. It is possible to approximate these rewards, as in the tabular case, by considering the ensemble of critics as samples from the value distribution. If we focus only on estimating the positive component of the `exact-ube` estimate, i.e., the local uncertainty defined by Zhou et al. (2020), then a sample-based approximation is given by

$$\hat{w}(s,a) = \mathbb{V}_i\left[\big\{\bar{Q}(s_i', a_i')\big\}_{i=1}^N\right], \tag{2.12}$$

where $s_i' \sim p_i(\cdot \mid s,a)$. While this approach is sensible from our theory perspective and has lead to promising results in our previous work (Luis et al., 2023a), it has two main shortcomings in practice: (*i*) it can be computationally intensive to estimate the rewards and (*ii*) the magnitude of the rewards is typically low, even if the individual critics have largely different estimated values. The latter point is illustrated in Figure 2.4: the term $\hat{w}(s,a)$ captures the local variance of the average value function, which would be small if the function is relatively flat around (s,a) or if the dynamics model ensemble yields similar forward predictions starting from (s,a). Empirically, we found that across many environments the average magnitude of $\hat{w}(s,a)$ is indeed small (e.g., $\sim 10^{-3}$), which makes training a U-net challenging mainly due to vanishing gradients from the *softplus* layer. We alleviate both shortcomings via a simple proxy uncertainty reward:

$$\hat{w}_{\text{ub}}(s,a) = \mathbb{V}_i\left[\big\{Q_i(s,a)\big\}_{i=1}^N\right], \tag{2.13}$$

which is the sample-based approximation of the value variance. We denote this estimate `upper-bound` (thus, the subscript "ub" in (2.13)), since in the limit of infinite samples from the value distribution, solving a UBE with rewards $\hat{w}_{\text{ub}}(s,a)$ results in an upper bound on the value variance at (s,a).

The proxy rewards $\hat{w}_{\text{ub}}(s,a)$ capture explicitly the value ensemble disagreement rather than local variations of the average value, which empirically results in larger rewards being propagated through the UBE. Moreover, the proxy reward calculation requires only one forward pass through the critic ensemble, without need for forward predictions with the dynamics model as for $\hat{w}(s,a)$.

Variance estimate. Similar to critic training, we model the variance estimate $\hat{U}^\pi$ with a neural network, denoted U-net, parameterized by φ and trained to minimize the UBE residual

$$\mathcal{L}(\varphi) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}_{\text{model}}}\left[\big(z - U(s,a;\varphi)\big)^2\right], \tag{2.14}$$

with targets $z = \gamma^2 \hat{w}_{\text{ub}}(s,a) + \gamma^2 U(s', a'; \bar{\varphi})$ and target parameters $\bar{\varphi}$ updated like in regular critics. Since we interpret the output of the network as predictive variances, we use a *softplus* output layer to guarantee

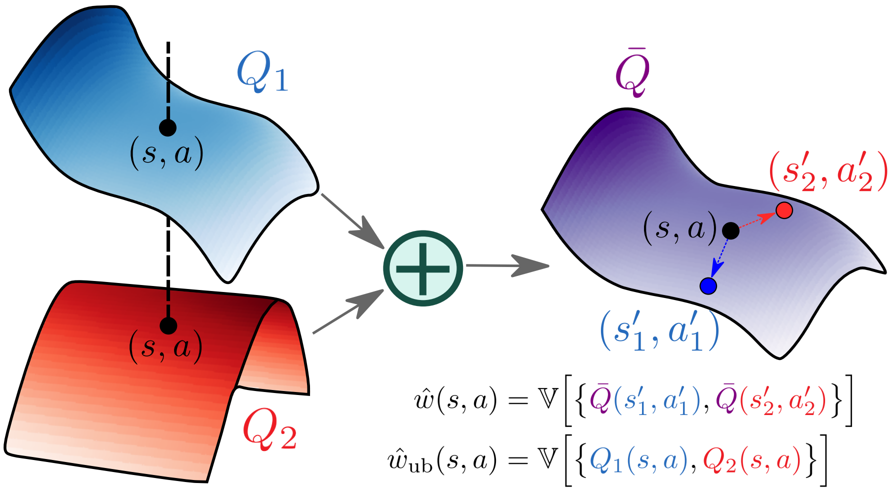

Q_1
(s, a)
(s, a)
Q_2
$\bar{Q}$
(s_2', a_2')
(s, a)
(s_1', a_1')
$\hat{w}(s,a) = \mathbb{V}\Big[\{\bar{Q}(s_1', a_1'), \bar{Q}(s_2', a_2')\}\Big]$
$\hat{w}_{\text{ub}}(s,a) = \mathbb{V}\Big[\{Q_1(s,a), Q_2(s,a)\}\Big]$

Figure 2.4.: Illustrative example of uncertainty rewards. **(Left)** ensemble of two value functions $\{Q_1, Q_2\}$. **(Right)** corresponding mean value function $\bar{Q}$. The theory prescribes estimating the term in (2.5), denoted $\hat{w}(s,a)$, which captures local variability of $\bar{Q}$ around (s,a). Empirically, $\hat{w}(s,a)$ can be small despite large differences in individual members of the value ensemble, e.g., because $\bar{Q}$ is relatively flat around (s,a). We propose the proxy uncertainty reward $\hat{w}_{\text{ub}}(s,a)$ which directly captures variability across the value ensemble and is less computationally expensive (no dynamics model forward pass).

non-negative values. Moreover, we apply a symlog transformation to the UBE targets z, as proposed by Hafner et al. (2023), which helps the U-net converge to the target values more easily. Namely, the U-net is trained to predict the symlog transform of the target values z, defined as $\text{symlog}(z) = \text{sign}(z)\log\big(|z|+1\big)$. To retrieve the U-values, we apply the inverse transform $\text{symexp}(z) = \text{sign}(z)(\exp\big(|z|\big) - 1)$ to the output of the U-net.

Actor. The stochastic policy is represented as a neural network with parameters ϕ, denoted by π_ϕ. The policy's objective is derived from SAC, where in addition to entropy regularization, we include the predicted standard deviation of values for uncertainty-aware optimization.

$$\mathcal{L}(\phi) = \mathbb{E}_{s\sim\mathcal{D}_{\text{model}}}\Bigg[\mathbb{E}_{a\sim\pi_\phi}\Big[\bar{Q}(s,a) + \lambda\sqrt{U(s,a)} - \alpha\log\pi_\phi(a \mid s)\Big]\Bigg]. \tag{2.15}$$

Online vs offline optimization. With QU-SAC we aim to use largely the same algorithm to tackle both online and offline problems. Beyond differences in hyperparameters, the only algorithmic change in QU-SAC is that for offline optimization we modify the data used to train the actor, critic and U-net to also include data from the offline dataset (an even 50/50 split between offline and model-generated data in our case), which is a standard practice in offline model-based RL (Rigter et al., 2022; Yu et al., 2020, 2021; Jeong et al., 2023).

2.5. QU-SAC Online Implementation

In this section, we provide details regarding the online implementation of QU-SAC.

Algorithm 2 QU-SAC (online)

```
1: Initialize policy $\pi_\phi$, predictive model $p_\theta$, critic ensemble $\{Q_i\}_{i=1}^N$, uncertainty net $U_\psi$ (optional), environment dataset $\mathcal{D}$, model datasets $\mathcal{D}_{\text{model}}$ and $\{\mathcal{D}_i\}_{i=1}^N$.
2: global step $\leftarrow 0$
3: for episode $t = 0, \ldots, T-1$ do
4:   for $E$ steps do
5:     if global step % $F == 0$ then
6:       Train model $p_\theta$ on $\mathcal{D}$ via maximum likelihood
7:       for $M$ model rollouts do
8:         Perform $k$-step model rollouts starting from $s \sim \mathcal{D}$; add to $\mathcal{D}_{\text{model}}$ and $\{\mathcal{D}_i\}_{i=1}^N$
9:     Take action in environment according to $\pi_\phi$; add to $\mathcal{D}$
10:    for $G$ gradient updates do
11:      Update $\{Q_i\}_{i=1}^N$ with mini-batches from $\{\mathcal{D}_i\}_{i=1}^N$, via SGD on (2.11)
12:      (Optional) Update $U_\psi$ with mini-batches from $\mathcal{D}_{\text{model}}$, via SGD on (2.14)
13:      Update $\pi_\phi$ with mini-batches from $\mathcal{D}_{\text{model}}$, via SGD on (2.15)
14:   global step $\leftarrow$ global step $+1$
```

We build QU-SAC on top of MBPO (Janner et al., 2019) following Algorithm 2. The main differences with the original implementation are as follows:

- In Line 8, we perform a total of $N+1$ k-step rollouts corresponding to both the model-randomized and model-consistent rollout modalities. The original MBPO only executes the former to fill up $\mathcal{D}_{\text{model}}$.
- In Line 11, we update the ensemble of Q-functions on the corresponding model-consistent buffer. MBPO trains twin critics (as in SAC) on mini-batches from $\mathcal{D}_{\text{model}}$.
- In Line 12, we update the U-net for the UBE-based variance estimation methods.
- In Line 13, we update π_ϕ by maximizing the uncertainty-aware Q-values. MBPO maximizes the minimum of the twin critics (as in SAC). Both approaches include an entropy maximization term.

The main hyperparameters for our experiments are included in Table A.1. Further implementation details are now provided.

Model learning. We leverage the `mbrl-lib` Python library from Pineda et al. (2021) and train an ensemble of N probabilistic neural networks. We use the default MLP architecture with four layers of size 200 and SiLU activations. The networks predict delta states, $\Delta = s' - s$, and receive as input state-action pairs. We use the default initialization of the network provided by the library, which samples weights from a truncated Gaussian distribution, however we found it helpful to increase by a factor of 2.0 the standard deviation of the truncated Gaussian; a wider distribution of weights allows for more diverse dynamic models at the beginning of training.

Model-generated buffers. The capacity of the model-generated buffers $\mathcal{D}_{\text{model}}$ and $\{\mathcal{D}^i_{\text{model}}\}_{i=1}^N$ is computed as $k \times M \times F \times \Delta$, where Δ is the number of model updates before entirely overwriting the buffers. Larger values of this parameter allows for more off-policy (old) data to be stored and sampled for training.

SAC specifics. Our SAC implementation is based on the open-source repository `https://github.com/pranz24/pytorch-soft-actor-critic`, as done by `mbrl-lib`. For all our experiments, we use the automatic entropy tuning flag that adaptively modifies the entropy gain α based on the stochasticity of the policy.

Algorithm 3 QU-SAC (offline)

```
1: Initialize policy $\pi_\phi$, predictive model $p_\theta$, critic ensemble $\{Q_i\}_{i=1}^N$, uncertainty net $U_\psi$ (optional), offline
   dataset $\mathcal{D}$, model datasets $\mathcal{D}_{\text{model}}$ and $\{\mathcal{D}_i\}_{i=1}^N$.
2: Train model $p_\theta$ on $\mathcal{D}$ via maximum likelihood
3: for steps $g = 0, \dots, G-1$ do
4:    if g % $F == 0$ then
5:       for $L$ model rollouts do
6:          Perform $k$-step model rollouts starting from $s \sim \mathcal{D}$; add to $\mathcal{D}_{\text{model}}$ and $\{\mathcal{D}_i\}_{i=1}^N$
7:    Update $\{Q_i\}_{i=1}^N$ with mini-batches from $\{\mathcal{D} \cup \mathcal{D}_i\}_{i=1}^N$, via SGD on (2.11)
8:    (Optional) Update $U_\psi$ with mini-batches from $\mathcal{D} \cup \mathcal{D}_{\text{model}}$, via SGD on (2.14)
9:    Update $\pi_\phi$ with mini-batches from $\mathcal{D} \cup \mathcal{D}_{\text{model}}$, via SGD on (2.15)
```

2.6. QU-SAC Offline Implementation

In this section, we provide further details regarding the use of QU-SAC for offline optimization.

We modify the online version of QU-SAC described in Algorithm 2 to reflect the execution flow of offline optimization, which we present in Algorithm 3. Beyond the algorithmic changes, we now list the main implementation details differing from the online implementation of QU-SAC:

Model learning. The only difference w.r.t. the online setting is that the we normalize the state-action inputs to the model, where the normalization statistics are calculated based on the offline dataset $\mathcal{D}$.

Data mixing. In Lines 7-9, we highlight that, in contrast to the online setting, the mini-batches used to train the critic, actor and U-net mix both model-generated and offline data. In particular, we use a fixed 50/50 split between these two data sources.

2.7. Experiments

In this section, we empirically evaluate the performance of our risk-aware policy optimization scheme (2.10) in various problems and compare against related baselines.

2.7.1. Baselines

In Algorithm 1, we consider different implementations of the `qvariance` method to estimate $\hat{U}(s, a)$: `ensemble-var` directly uses the sample-based approximation $\hat{w}_{\text{ub}}(s, a)$ in (2.13); `pombu` uses the solution to the UBE (2.6); `exact-ube` uses the solution to our proposed UBE (2.7); and `upper-bound` refers to the solution of the UBE with the modified rewards (2.13). We also compare against not using any form of uncertainty quantification, which we refer to as `ensemble-mean`.

Additionally, in tabular problems we include PSRL by Osband et al. (2013) as a baseline since it typically outperforms recent OFU-based methods (O'Donoghue, 2021; Tiapkin et al., 2022). We also include MBPO (Janner et al., 2019) and MOPO (Yu et al., 2020) as baselines for online and offline problems, respectively.

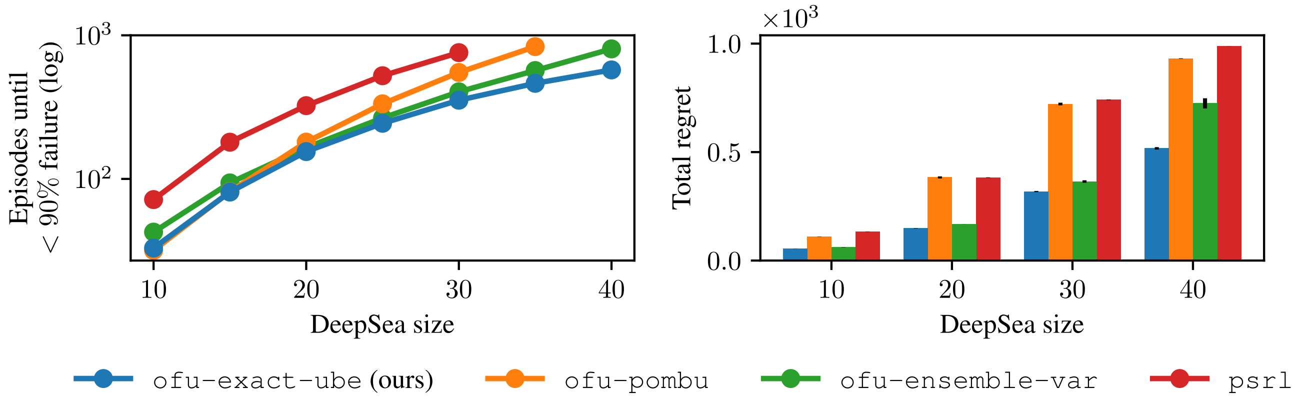


Figure 2.5.: Performance in the *DeepSea* benchmark. Lower values in plots indicate better performance. (Left) Learning time is measured as the first episode where the sparse reward has been found at least in 10% of episodes so far. (Right) Total regret is approximately equal to the number of episodes where the sparse reward was not found. Results represent the average over 5 random seeds, and vertical bars on total regret indicate the standard error. Our variance estimate achieves the lowest regret and best scaling with problem size.

2.7.2. Gridworld Exploration Benchmark

We evaluate the tabular implementation in two grid-world environments where exploration is key to find the optimal policy: *DeepSea* and *7-room*.

Tabular Implementation Details

In this section, we provide more details about the tabular implementation of Algorithm 1.

Model learning. For the transition function we use a prior Dirichlet$(1/\sqrt{S})$ and for rewards a standard normal $\mathcal{N}(0,1)$, as done by O'Donoghue et al. (2019). The choice of priors leads to closed-form posterior updates based on state-visitation counts and accumulated rewards. We add a terminal state to our modeled MDP in order to compute the values in closed-form via linear algebra.

Accelerating learning. For the *DeepSea* benchmark we accelerate learning by imagining each experienced transition (s, a, s', r) is repeated L times, as initially suggested by Osband et al. (2019) (see footnote 9), although we scale the number of repeats with the size of the MDP. Effectively, this strategy forces the MDP posterior to shrink faster, thus making all algorithms converge in fewer episodes. The same strategy was used for all the methods evaluated in the benchmark.

Policy optimization. All tested algorithms (PSRL and OFU variants) optimize the policy via policy iteration, where we break ties at random when computing the argmax, and limit the number of policy iteration steps to 40.

Hyperparameters. Unless noted otherwise, all tabular RL experiments use a discount factor $\gamma = 0.99$, an exploration gain $\lambda = 1.0$ and an ensemble size $N = 5$.

Uncertainty reward clipping. For *DeepSea* we clip uncertainty rewards with $u_{\min} = -0.05$ and for the 7-room environment we keep $u_{\min} = 0.0$.

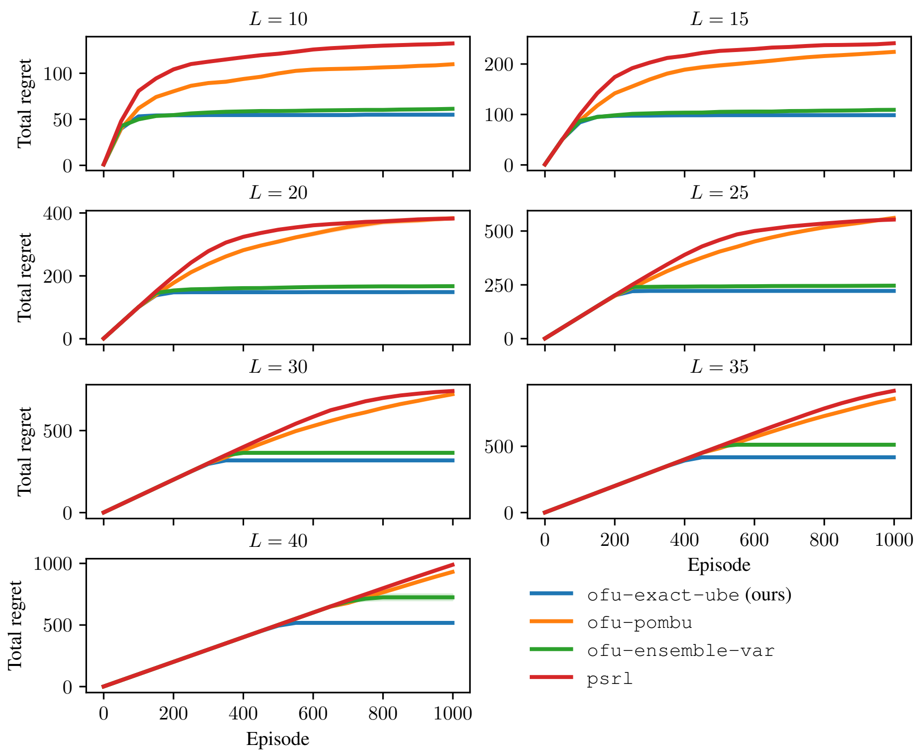


Figure 2.6.: Extended results for the *DeepSea* experiments shown in Figure 2.5. We report the average (solid line) and standard error (shaded region) over 5 random seeds.

DeepSea

First proposed by Osband et al. (2019), this environment tests the agent's ability to explore over multiple time steps in the presence of a deterrent. It consists of an $L \times L$ grid-world MDP, where the agent starts at the top-left cell and must reach the lower-right cell. The agent decides to move left or right, while always descending to the row below. We consider the deterministic version of the problem, so the agent always transitions according to the chosen action. Going left yields no reward, while going right incurs an action cost (negative reward) of $0.01/L$. The bottom-right cell yields a reward of 1, so that the optimal policy is to always go right. As the size of the environment increases, the agent must perform sustained exploration in order to reach the sparse reward.

The experiment consists on running each method for 1000 episodes and five random seeds, recording the total regret and "learning time", defined as the first episode where the rewarding state has been found at least in 10% of the episodes so far (O'Donoghue, 2021). For this experiment, we found that using $u_{\min} = -0.05$ improves the performance of our method: since the underlying MDP is acyclic, propagating negative uncertainty rewards is consistent with our theory.

Figure 2.5 (left) shows the evolution of learning time as L increases. Our method achieves the lowest learning time and best scaling with problem size. Notably, all the OFU-based methods learn faster than PSRL, a strong argument in favour of using the variance of value functions to guide exploration. Figure 2.5 (right) shows that our approach consistently achieves the lowest total regret across all values of L. This empirical evidence indicates that the solution to our UBE can be integrated into common exploration techniques like UCB to serve as an effective uncertainty signal. Moreover, our method significantly improves peformance over `pombu`, highlighting the relevance of our theory results. We include the corresponding training curves in Figure 2.6.

Uncertainty rewards ablation. Our theory prescribes equivalent expressions for the uncertainty rewards under the assumptions. However, since in practice the assumptions do not generally hold, the expressions are no longer equivalent. We evaluate the performance in the *DeepSea* benchmark for these different definitions of the uncertainty rewards:

- `exact-ube_1`:
$$u(s,a) = \mathbb{V}_{a',s' \sim \pi, \bar{p}}\big[\bar{q}^{\pi}(s',a')\big] - \mathbb{E}_P\Big[\mathbb{V}_{a',s' \sim \pi, P}\big[Q^{\pi}(s',a')\big]\Big]$$

- `exact-ube_2`:
$$u(s,a) = \mathbb{V}_P\left[\sum_{a',s'} \pi(a' \mid s')P(s' \mid s,a)\bar{q}^{\pi}(s',a')\right] - \mathbb{E}_P\Big[\mathbb{V}_{a',s' \sim \pi, P}\big[Q^{\pi}(s',a')\big] - \mathbb{V}_{a',s' \sim \pi, P}\big[\bar{q}^{\pi}(s',a')\big]\Big]$$

- `exact-ube_3` (labeled `exact-ube` in all other plots):
$$u(s,a) = \mathbb{V}_P\left[\sum_{a',s'} \pi(a' \mid s')P(s' \mid s,a)\bar{q}^{\pi}(s',a')\right] - \mathbb{E}_P\Big[\mathbb{V}_{a',s' \sim \pi, P}\big[Q^{\pi}(s',a') - \bar{q}^{\pi}(s',a')\big]\Big]$$

Figure 2.7 shows the results for the *DeepSea* benchmark comparing the three uncertainty signals. Since the assumptions are violated in the practical setting, the three signals are no longer equivalent and result in slightly different uncertainty rewards. Still, when integrated into Algorithm 1, the performance in terms of learning time and total regret is quite similar. We select `exact-ube_3` as the default estimate for all other experiments.

Ensemble size ablation. The ensemble size N is one important hyperparameter for all the OFU-based methods. We perform additional experiments in *DeepSea* for different values of N, keeping all other hyperparameters fixed and with sizes $L = \{20, 30\}$. The results in Figure 2.8 show that our method achieves lower total regret across the different ensemble sizes. For `ensemble-var`, performance increases for larger ensembles. These results suggest that the sample-based approximation of our uncertainty rewards is not very sensitive to the number of samples and achieve good performance even for $N = 2$.

Exploration gain ablation. Another important hyperparameter for OFU-based methods is the exploration gain λ, controlling the magnitude of the optimistic values optimized via policy iteration. We perform an ablation study over λ, keeping all other hyperparameters fixed and testing for *DeepSea* sizes $L = \{20, 30\}$. Figure 2.9 shows the total regret for OFU methods over increasing gain. Unsurprisingly, as we increase λ, the total regret of all the methods increases, but overall `exact-ube` achieves the best performance.

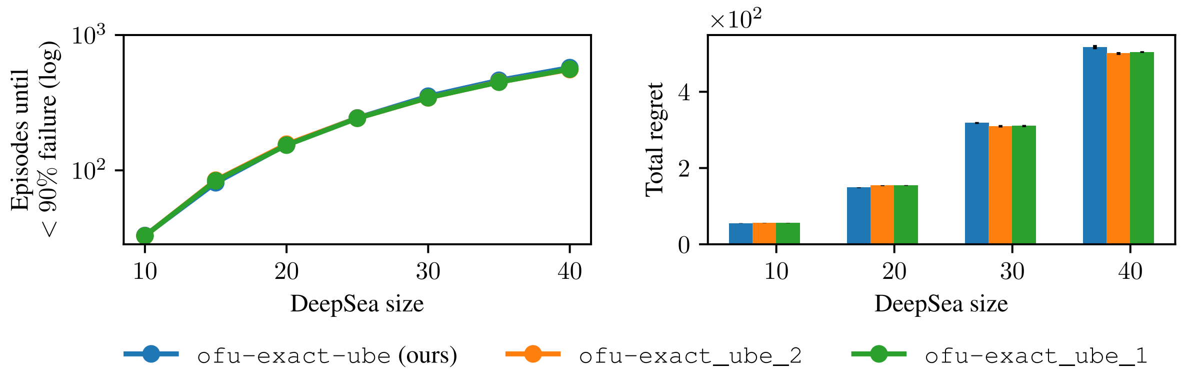


Figure 2.7.: Ablation study on *DeepSea* exploration for different estimates of `exact-ube`. Results represent the average over 5 random seeds, and vertical bars on total regret indicate the standard error.

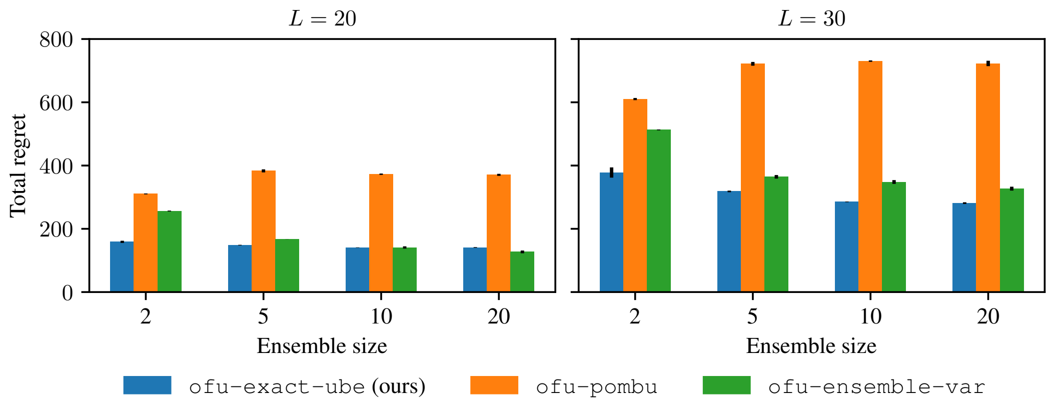


Figure 2.8.: Ablation study over ensemble size N on the *DeepSea* environment.

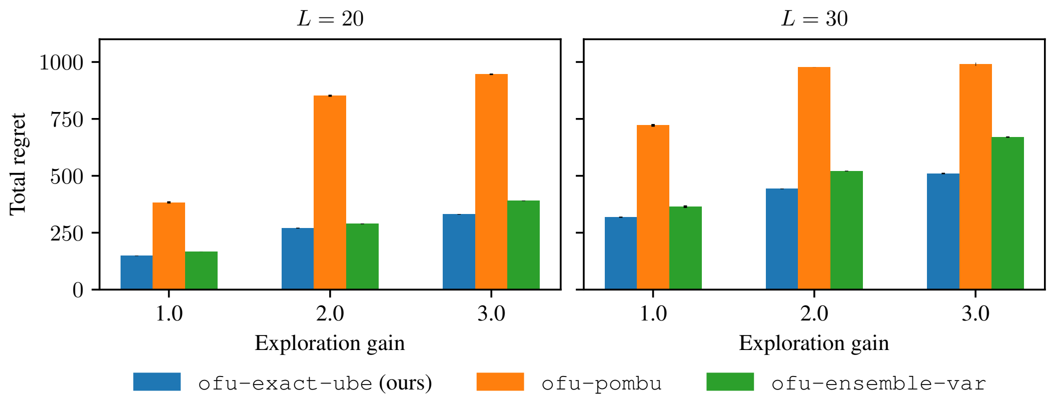


Figure 2.9.: Ablation study over exploration gain λ on the *DeepSea* environment.

7-room

As implemented by Domingues et al. (2021), the 7-room environment consists of seven connected rooms of size 5×5. The agent starts in the center of the middle room and an episode lasts 40 steps. The possible actions are up-down-left-right and the agent transitions according to the selected action with probability 0.95, otherwise it lands in a random neighboring cell. The environment has zero reward everywhere except two small rewards at the start position and in the left-most room, and one large reward in the right-most room. Unlike *DeepSea*, the underlying MDP for this environment contains cycles, so it evaluates our method beyond the theoretical assumptions. In Figure 2.10, we show the regret curves over 5000 episodes. Our method achieves the lowest regret, which is remarkable considering recent empirical evidence favoring PSRL over OFU-based methods in these type of environments (Tiapkin et al., 2022). The large gap between `ensemble-var` and the UBE-based methods is due to overall larger variance estimates from the former, which consequently requires more episodes to reduce the value uncertainty.

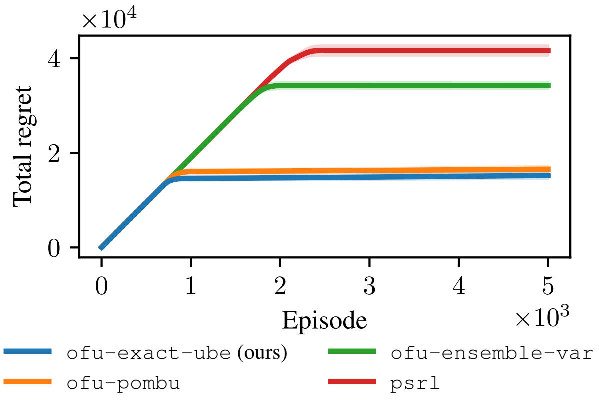


Figure 2.10.: Total regret curve for the 7-room environment. Lower regret is better. Results are the average (solid lines) and standard error (shaded regions) over 10 random seeds. Our method achieves the lowest regret, significantly outperforming PSRL.

2.7.3. DeepMind Control Suite - Exploration Benchmark

In this section, we evaluate the performance of QU-SAC for online exploration in environments with continuous state-action spaces. In Appendix A.3 we list specific hyperparameters in Table A.1.

We test the exploration capabilities of QU-SAC on a subset of environments from the DeepMind Control (DMC) suite (Tunyasuvunakool et al., 2020) with a sparse reward signal. Additionally, we modify the environments' reward signal to include a cost proportional to the squared norm of the action taken by the agent. Namely,

$$\texttt{action\_cost} = \rho \sum_{i=1}^{|\mathcal{A}|} a_i^2 \tag{2.16}$$

where ρ is an environment specific multiplier, a_i is the i-th component of the action vector and $|\mathcal{A}|$ is the size of the action space. For `acrobot`, `reacher-hard` and `cartpole-swingup` we use $\rho = 0.01$; for `pendulum` and `point-mass` we use $\rho = 0.05$; and lastly, for `ball-in-cup` we use $\rho = 0.2$.

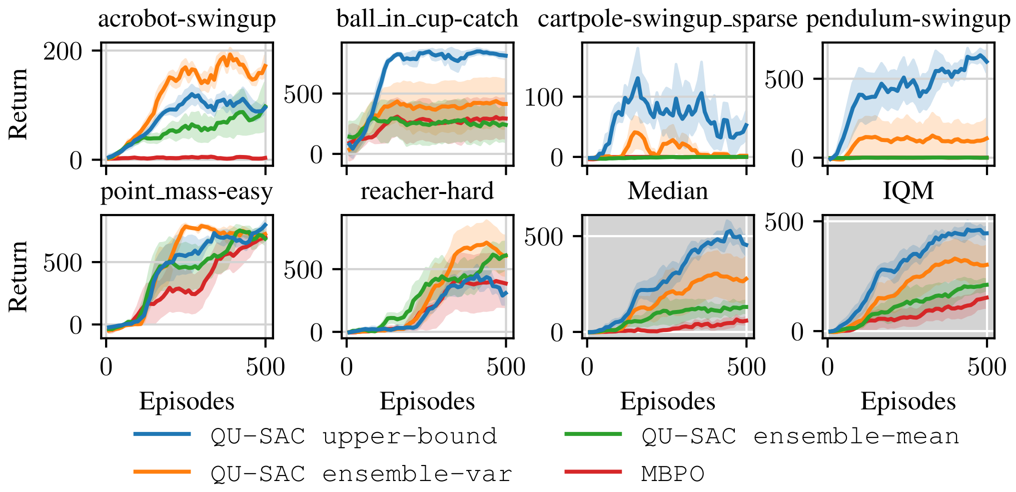


Figure 2.11.: DeepMind Control Suite Benchmark smoothened learning curves over 500 episodes (500K environment steps). We report the mean (solid) and standard error (shaded region) over five random seeds. QU-SAC with the `upper-bound` variance estimate outperforms the baselines in 4/6 environments and has the best overall performance.

Action costs are relevant for energy-constrained systems where the agent must learn to maximize the primary objective while minimizing the actuation effort. However, the added negative reward signal may inhibit exploration and lead to premature convergence to sub-optimal policies. In this context, we want to compare the exploration capabilities of QU-SAC with the different variance estimates.

In Figure 2.11 we plot the performance of all baselines in our exploration benchmark after 500 episodes (or equivalently, 500K environment steps). In addition to individual learning curves, we aggregate performance across all environments and report the median and inter-quartile mean (IQM) (Agarwal et al., 2021). The results highlight that QU-SAC with our proposed `upper-bound` variance estimate offers the best overall performance. The pendulum swingup environment is a prime example of a task where the proposed approach excels: greedily optimizing for mean values, like MBPO and `ensemble-mean`, does not explore enough to observe the sparse reward; `ensemble-var` improves performance upon the greedy approach, but does not work consistently across random seeds unlike `upper-bound`. In this case, the stronger exploration signal afforded by propagating uncertainty through the U-net is key to maintain exploration despite low variability on the critic ensemble predictions.

2.7.4. D4RL Offline Benchmark

In this section, we evaluate the performance of QU-SAC for offline RL in the Mujoco (Todorov et al., 2012) datasets from the D4RL benchmark (Fu et al., 2020). In Appendix A.3 we list specific hyperparameters in Table A.2.

The core idea behind QU-SAC for offline optimization is to leverage the predicted value uncertainty for conservative (pessimistic) policy optimization. This simply involves fixing $\lambda < 0$ to downweight values depending on their predicted uncertainty. In addition to uncertainty-based pessimism, prior work proposed SAC-M (An et al., 2021) which uses an ensemble of M critics and imposes conservatism by taking the minimum of the ensemble prediction as the value estimate. A key question we want to address with our

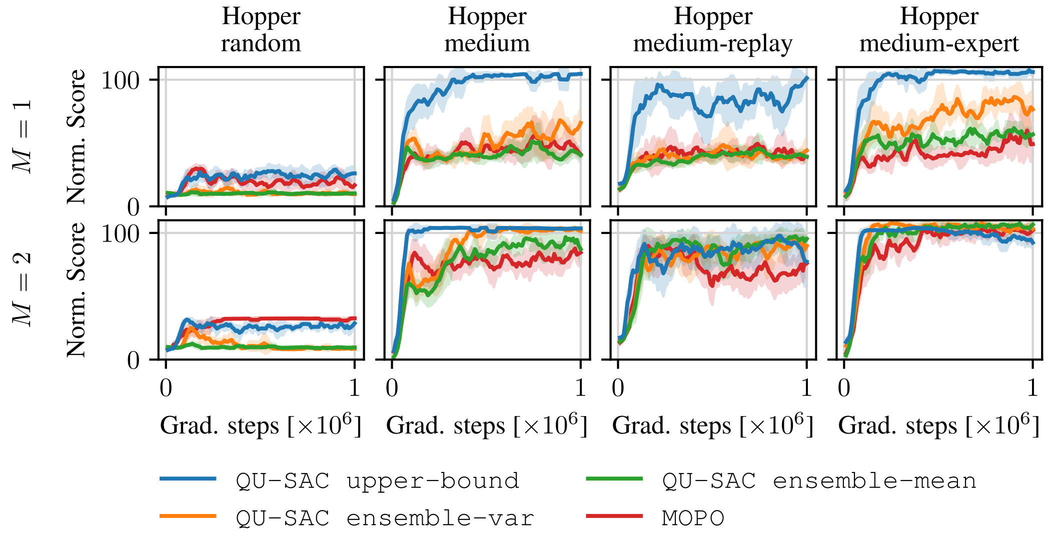


Figure 2.12.: D4RL learning curves for Hopper datasets, smoothened by a moving average filter. We report the mean (solid) and standard deviation (shaded region) over five random seeds of the average normalized score over 10 evaluation episodes. We use $M = 1$ for all baselines on the top row plots and $M = 2$ on the bottom row. QU-SAC with the `upper-bound` variance estimate provides the most consistent performance across both values of M.

experiments is whether pure uncertainty-based pessimism is enough to avoid out-of-distribution overestimation in offline RL.

In order to provide an empirical answer, we augment QU-SAC with SAC-M by training M critics for each of the N dynamics models. The result is an ensemble of $N \times M$ critics, labelled as Q_{ij} for $i = \{1, \dots, N\}$, $j = \{1, \dots, M\}$. Each subset of M critics is trained using clipped Q-learning (Fujimoto et al., 2018) as in SAC-M, where the i-th critic prediction is simply defined as $Q_i(s, a) = \min_j Q_{ij}(s, a)$. The mean critic prediction is redefined as the average over clipped Q-values, $\bar{Q}(s, a) = 1/N \sum_{i=1}^{N} \min_j Q_{ij}(s, a)$. If $M = 1$ we recover the original QU-SAC which only uses variance prediction as a mechanism for conservative optimization. Note that MOPO fixes $M = 2$, which means it combines uncertainty and clipped-based conservatism by default; we re-implemented MOPO in order to allow for arbitrary M. In this context, our key question becomes: can any of the methods perform well with $M = 1$, i.e., only using uncertainty-based pessimism?

MOPO details. In order to conduct a fair comparison between MOPO and QU-SAC, we implement MOPO in our codebase so that it shares the same core components as our QU-SAC implementation. After initial testing of our MOPO implementation, we found that using an uncertainty penalty of $\lambda = 1.0$ worked well across datasets. Note that our implementation of MOPO (labeled MOPO$\star$ in Table 2.3) significantly outperforms the scores reported by Bai et al. (2022), which were obtained by running the original codebase by Yu et al. (2020) but on the v2 datasets from D4RL.

We conduct experiments in D4RL (version v2) datasets for three environments (Hopper, HalfCheetah and Walker2D) and four tasks each (random, medium, medium-replay and medium-expert) for a total of 12 datasets. For each dataset, we pre-train an ensemble of dynamics models and then run offline optimization for 1M gradient steps. In Figure 2.12 we present the results for the Hopper datasets using $M = \{1, 2\}$. In the pure uncertainty-based pessimism setting ($M = 1$), QU-SAC with the `upper-bound` variance estimate

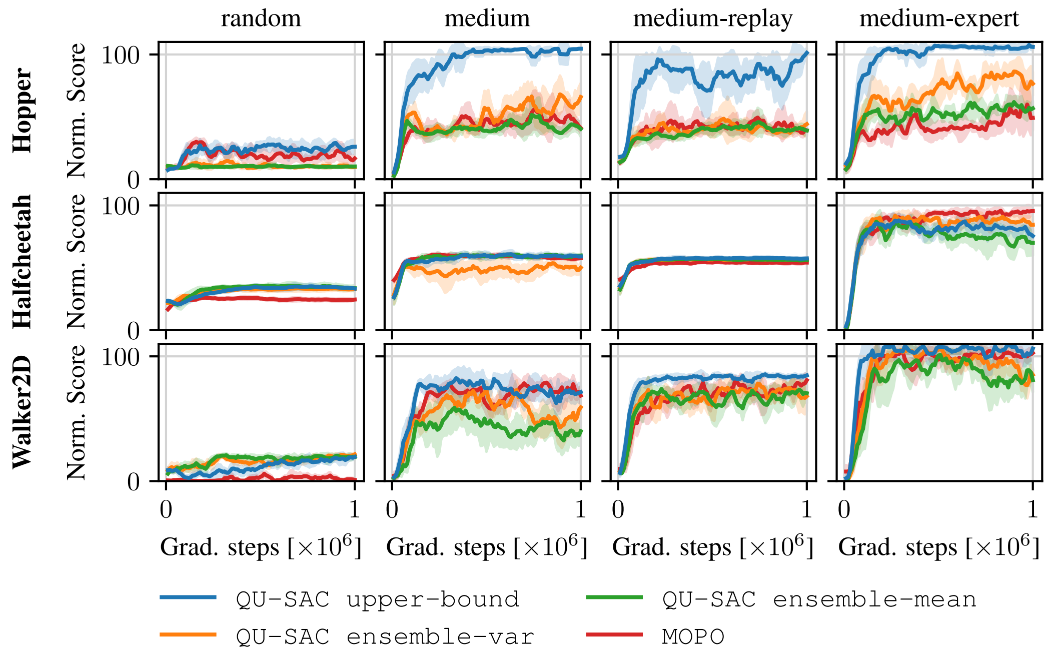


Figure 2.13.: D4RL smoothened learning curves for $M = 1$. We report the mean and standard deviation over five random seeds of the average normalized score over 10 evaluation episodes.

obtains the best performance by a wide margin. Qualitatively, the effect of supplementing `upper-bound` with clipped Q-learning ($M = 2$) is more stable performance rather than a significant score improvement, unlike most other baselines that do improve substantially. These results suggest that proper uncertainty quantification might be sufficient for offline learning, without relying on additional mechanisms to combat out-of-distribution biases such as clipped Q-learning.

In Figures 2.13 and 2.14 we include all the learning curves for $M = \{1, 2\}$, respectively. For each run, we report the average normalized score over 10 evaluation episodes. These averaged scores are then also averaged over five independent random seeds to obtain the reported learning curves. In Table 2.2 we report the associated final scores after 1M gradient steps.

We observe that for $M = 2$ `upper-bound` has lower overall performance than `ensemble-var`. We believe this difference in performance is largely due to using a fixed value of $\lambda = -1.0$ for all experiments. Since using $M = 2$ alread acts a strong regularizer in the offline setting, `upper-bound` would likely benefit from using a lower magnitude λ given the (empirically) larger uncertainty estimates compared to `ensemble-var`.

Table 2.2.: D4RL scores after 1M gradient steps. We report the mean and standard deviation over five random seeds of the average normalized score across 10 evaluation episodes. We highlight the highest mean scores for each value of M.

		$M=1$				$M=2$			
		MOPO⋆	e-mean	e-var	u-bound	MOPO⋆	e-mean	e-var	u-bound
Random	HalfCheetah	24.8±0.7	33.6 ±3.8	33.0±1.7	33.4±1.2	25.9±1.4	32.4±1.8	34.8 ±1.0	30.2±1.5
	Hopper	20.7±9.2	10.3±0.8	9.3±1.8	28.3 ±8.3	32.6 ±0.2	9.8±1.1	8.7±0.8	31.5±0.2
	Walker2D	0.5±0.3	20.3±4.4	21.9 ±1.0	18.2±5.0	1.0±1.9	20.5±2.3	21.7 ±0.1	21.7 ±0.1
Medium	HalfCheetah	57.7±1.5	60.6 ±1.4	58.5±2.9	59.7±2.5	60.6±2.4	60.3±0.8	63.7 ±2.3	60.6±1.7
	Hopper	35.4±4.1	41.4±8.8	75.3±24.1	104.7 ±1.0	81.3±15.7	78.8±19.8	102.0±3.8	103.5 ±0.2
	Walker2D	56.9±25.8	58.3±17.9	57.3±19.9	67.8 ±14.4	85.3±1.3	88.3±1.2	88.8 ±0.9	86.5±0.7
Medium Replay	HalfCheetah	53.5±2.0	56.2±1.5	57.3±1.2	57.5 ±0.9	55.7±0.9	58.9 ±0.7	58.4±0.7	58.9 ±1.3
	Hopper	36.0±2.7	38.1±4.3	42.3±8.4	102.0 ±1.2	69.0±27.0	100.3±3.6	102.9 ±0.5	86.2±20.9
	Walker2D	88.2 ±5.4	75.8±13.6	77.9±13.3	84.8±2.5	83.1±5.0	84.1 ±1.4	82.4±2.9	76.8±0.6
Medium Expert	HalfCheetah	98.0 ±3.6	68.6±17.7	86.9±17.4	74.3±16.8	95.0±1.7	99.5 ±2.4	99.5 ±1.9	99.1±2.5
	Hopper	47.6±8.3	56.3±14.9	65.6±16.1	107.0 ±1.2	104.5±7.7	106.9 ±3.0	102.1±12.6	93.8±10.4
	Walker2D	106.2±1.2	65.9±35.6	83.9±21.3	106.8 ±5.1	107.7±0.8	108.4 ±0.5	107.9±0.4	93.7±25.6
	Average	52.1	48.8	55.8	70.4	66.8	70.7	72.7	70.2
	IQM	47.9	51.8	59.4	74.4	72.5	78.3	82.5	77.1

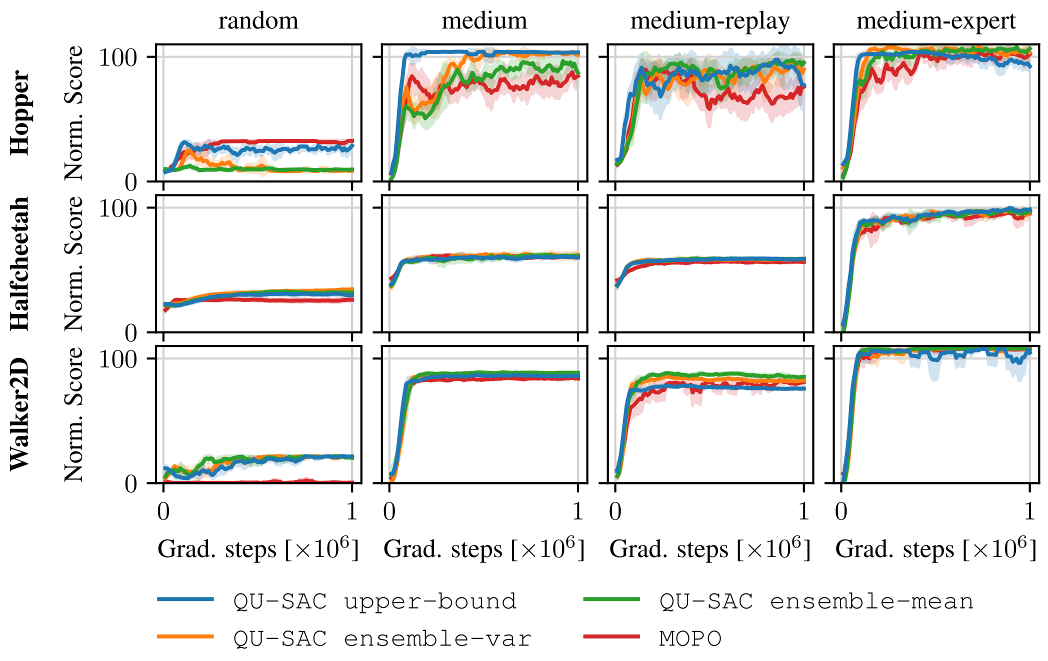


Figure 2.14.: D4RL smoothened learning curves for $M=2$. We report the mean and standard deviation over five random seeds of the average normalized score over 10 evaluation episodes.

Table 2.3.: D4RL final normalized scores for model-based RL algorithms. The highest average scores are highlighted in light blue. MOPO$\star$ corresponds to our own implementation of the algorithm by Yu et al. (2020), while QU-SAC utilizes the `upper-bound` variance estimate and $M = 2$. For MOPO$\star$ and QU-SAC, we report the mean and standard deviation over five random seeds. The scores for the original MOPO results are as reported by Bai et al. (2022). We take the results for COMBO (Yu et al., 2021), RAMBO (Rigter et al., 2022) and CBOP (Jeong et al., 2023) from their corresponding papers.

		MOPO	MOPO$\star$	COMBO	RAMBO	CBOP	**QU-SAC**
Random	HalfCheetah	35.9±2.9	25.9±1.4	38.8±3.7	40.0 ±2.3	32.8±0.4	30.2±1.5
	Hopper	16.7±12.2	32.6 ±0.2	17.9±1.4	21.6±8.0	31.4±0.0	31.5±0.2
	Walker2D	4.2±5.7	1.0±1.9	7.0±3.6	11.5±10.5	17.8±0.4	21.7 ±0.1
Medium	HalfCheetah	73.1±2.4	60.6±2.4	54.2±1.5	77.6 ±1.5	74.3±0.2	60.7±1.7
	Hopper	38.3±34.9	81.3±15.7	97.2±2.2	92.8±6.0	102.6±0.1	103.5 ±0.2
	Walker2D	41.2±30.8	85.3±1.3	81.9±2.8	85.0±15.0	95.5 ±0.4	86.5±0.7
Medium Replay	HalfCheetah	69.2 ±1.1	55.7±0.9	55.1±1.0	68.9±2.3	66.4±0.3	58.9±1.3
	Hopper	32.7±9.4	69.0±27.0	89.5±2.8	96.6±7.0	104.3 ±0.4	86.2±20.9
	Walker2D	73.7±9.4	83.1±5.0	56.0±8.6	85.0±15.0	92.7 ±0.9	76.8±0.6
Medium Expert	HalfCheetah	70.3±21.9	95.0±1.7	90.0±5.6	93.7±10.5	105.4 ±1.6	99.1±2.5
	Hopper	60.6±32.5	104.5±7.7	111.1 ±2.9	83.3±9.1	111.6 ±0.2	93.8±10.4
	Walker2D	77.4±27.9	107.7±0.8	103.3±5.6	68.3±20.6	117.2 ±0.5	93.7±25.6
	Average	49.4	66.8	66.8	68.7	79.3	70.2
	IQM	52.6	72.5	71.1	78.0	89.3	77.1

In Table 2.3, we compare the final scores of QU-SAC (using `upper-bound` and $M = 2$) against recent model-based offline RL methods. While scores are typically lower than the state-of-the-art method CBOP (Jeong et al., 2023), our general-purpose method outperforms MOPO and is on-par with more recent and stronger model-based baselines like COMBO and RAMBO[3].

2.8. Conclusions

In this chapter, we derived an uncertainty Bellman equation whose fixed-point solution converges to the variance of values given a posterior distribution over MDPs. Our theory brings new understanding by characterizing the gap in previous UBE formulations that upper-bound the variance of values. We showed that this gap is the consequence of an over-approximation of the uncertainty rewards being propagated through the Bellman recursion, which ignore the inherent *aleatoric* uncertainty from acting in an MDP. Instead, our theory recovers exclusively the *epistemic* uncertainty due to limited environment data, thus serving as an effective exploration signal. The tighter variance estimate showed improved regret in typical tabular exploration problems.

Beyond tabular RL, we identified challenges on applying the UBE theory for uncertainty quantification and

[3]When $M = 1$, QU-SAC using `upper-bound` obtains an average score of 70.4 (IQM of 74.4) (see Table 2.2 in the supplementary material), which is also comparable to the reported performance of COMBO and RAMBO.

proposed a simple proxy uncertainty reward to overcome them. Based on this approximation, we introduced the Q-Uncertainty Soft Actor-Critic (QU-SAC) algorithm that can be used for both online and offline RL with minimal changes. For online RL, the proposed proxy uncertainty reward was instrumental for exploration in sparse reward problems. In offline RL, we demonstrate QU-SAC has solid performance without additional regularization mechanisms unlike other uncertainty quantification methods.

3. Value-Distributional Model-Based Reinforcement Learning

Quantifying uncertainty about a policy's long-term performance is important to solve sequential decision-making tasks. We study the problem from a model-based Bayesian reinforcement learning perspective, where the goal is to learn the posterior distribution over value functions induced by parameter (epistemic) uncertainty of the Markov decision process. Previous work restricts the analysis to a few moments of the distribution over values or imposes a particular distribution shape, e.g., Gaussians. Inspired by distributional reinforcement learning, we introduce a Bellman operator whose fixed-point is the value distribution function. Based on our theory, we propose Epistemic Quantile-Regression (EQR), a model-based algorithm that learns a value distribution function. We combine EQR with soft actor-critic (SAC) for policy optimization with an arbitrary differentiable objective function of the learned value distribution. Evaluation across several continuous-control tasks shows performance benefits with respect to both model-based and model-free algorithms. The code is available at `https://github.com/boschresearch/dist-mbrl`.

3.1. Introduction

Reinforcement learning (RL) tackles optimal decision-making in an unknown Markov Decision Process (MDP) (Sutton and Barto, 2018). Uncertainty is at the heart of the RL problem: on one hand, aleatoric uncertainty refers to the stochasticity in the MDP transitions and the RL agent's action selection; on the other hand, *epistemic* uncertainty appears due to lack of knowledge about the MDP. During policy evaluation, both sources of uncertainty induce a distribution of possible returns, which should be considered for policy optimization. For instance, in high-stakes applications like medical treatments, accounting for aleatoric noise is key towards training risk-averse policies (Chow et al., 2015; Keramati et al., 2020). Similarly, effective exploration can be achieved by proper handling of epistemic uncertainty (Deisenroth and Rasmussen, 2011; Curi et al., 2020).

Two paradigms have emerged to capture uncertainty in the predicted outcomes of a policy. First, *distributional* RL (Bellemare et al., 2017) models the aleatoric uncertainty about returns, due to the inherent noise of the decision process. In contrast, *Bayesian* RL (Ghavamzadeh et al., 2015) captures the epistemic uncertainty about the unknown *expected* return of a policy, denoted as the *value* function, due to incomplete knowledge of the MDP. As such, the distribution over outcomes from each perspective has fundamentally different meaning and utility. If we care about effective exploration of *unknown* (rather than stochastic) outcomes, then Bayesian RL is the appropriate choice of framework (Osband et al., 2019).

In this chapter, we focus on the Bayesian RL setting where a posterior distribution over possible MDPs induces a distribution over value functions. The posterior over values naturally models the epistemic uncertainty about the long-term performance of the agent, which is the guiding principle behind provably-efficient exploration (Strehl and Littman, 2008; Jaksch et al., 2010). An open question remains how to effectively model and learn the posterior distribution over value functions. We approach this problem by using tools from distributional RL in the Bayesian framework. The key idea is that, for time-inhomogeneous MDPs with a tabular representation, the value distribution follows a Bellman equation from which we can derive an iterative estimation algorithm that resembles methods from distributional RL. Based on this insight, we present a novel algorithm that uses a learned value distribution for policy optimization.

Our contribution. We introduce the value-distributional Bellman equation that describes the relationship between the value distributions over successive steps. Moreover, we show that the fixed-point of the associated Bellman operator is precisely the posterior value distribution. Then, leveraging tools from distributional RL, we propose a practical algorithm for learning the *quantiles* of the value distribution function. We propose Epistemic Quantile-Regression (EQR), a model-based policy optimization algorithm that learns a distributional value function. Finally, we combine EQR with soft actor-critic (SAC) to optimize a policy for any differentiable objective function of the learned value distribution (e.g., mean, exponential risk, CVaR, etc.)

3.1.1. Related work

Distributional RL. The treatment of the policy return as a random variable dates back to Sobel (1982), where it is shown that the higher moments of the return obeys a Bellman equation. More recently, distributional RL has emerged as a paradigm for modelling and utilizing the entire distribution of returns (Tamar et al., 2013; Bellemare et al., 2023), with real-world applications including guidance of stratospheric balloons (Bellemare et al., 2020) and super-human race-driving in simulation (Wurman et al., 2022). The distributional RL toolbox has expanded over the years with diverse distribution representations (Bellemare et al., 2017; Dabney et al., 2018b,a; Yang et al., 2019) and deeper theoretical understanding (Bellemare et al., 2019; Rowland et al., 2018; Lyle et al., 2019). In our core algorithm, we use quantile-regression (QR) by Dabney et al. (2018b) as a tool for learning the value, rather than return, distribution. Moreover, QR has been integrated with soft actor-critic (SAC) (Haarnoja et al., 2018) for improved performance (Wurman et al., 2022; Kuznetsov et al., 2020). At a high level, in this chapter we combine model learning with quantile-regression, which is then integrated with SAC for policy optimization.

Bayesian RL. Model-free approaches to Bayesian RL directly model the distribution over values, e.g., with normal-gamma priors (Dearden et al., 1998), Gaussian Processes (Engel et al., 2003) or ensembles of neural networks (Osband et al., 2016). Instead, model-based Bayesian RL represents uncertainty in the MDP dynamics, which must then be propagated to the value function. For instance, the PILCO algorithm by Deisenroth and Rasmussen (2011) learns a Gaussian Process model of the transition dynamics and integrates over the model's total uncertainty to obtain the expected values. In order to scale to high-dimensional continuous-control problems, Chua et al. (2018) use ensembles of probabilistic neural networks (NNs) to capture both aleatoric and epistemic uncertainty, as first proposed by Lakshminarayanan et al. (2017). Both approaches propagate model uncertainty during policy evaluation and improve the policy via greedy exploitation over this model-generated noise.

Closest to our problem setting are approaches that explicitly model the value distribution function or statistics thereof. The uncertainty Bellman equation (UBE) offers a framework to estimate the *variance* of the value distribution (O'Donoghue et al., 2018; Zhou et al., 2020; Luis et al., 2023a). Jorge et al. (2020) propose a principled backwards induction framework to estimate value distributions, with the caveat of assuming a Gaussian parameterization for practical implementation. Perhaps closest to our approach is the work by Dearden et al. (1999), which introduces a local sampling scheme that maintains a sample-based approximation of the value distribution, which is updated using a Bellman equation. While it does not assume a restrictive parametric form for the distribution, it ignores that samples from the value distribution at successive states are correlated through the Bellman equation; we make a similar approximation in our theory, see Section 3.3. In our work, rather than generating random samples of the value distribution, we keep track of a relevant set of statistics (Rowland et al., 2019), e.g., evenly spread quantiles, that have adequate coverage and representation power of the underlying distribution.

Mixed Approaches. Recent methods have combined distributional and model-based RL methods. Kastner et al. (2023) introduce the distributional model equivalence principle to train models that can plan optimally for risk-sensitive objectives. Moskovitz et al. (2021); Eriksson et al. (2022) aim to capture both sources of uncertainty by training an ensemble of return-distributional critics, where each critic models aleatoric uncertainty and the ensemble recovers epistemic uncertainty. Our approach is fundamentally different: we leverage tools from distributional RL to model the epistemic uncertainty around *expected* returns, i.e., we average over aleatoric noise. Moreover, our experiments show that our value representation with quantiles leads to substantial gains in performance over an ensemble of critics.

Uncertainty-Aware Policy Optimization. There exists a wide variety of policy optimization objectives that leverage epistemic uncertainty. Multi-model MDPs (MMDPs) (Steimle et al., 2021) consider a discrete distribution of MDPs and study the optimization of the average value under the MDP uncertainty. Solving exactly for the optimal policy is only possible for small MMDPs, but more recent methods can scale to larger problems (Su and Petrik, 2023). Robust MDPs optimize for risk-averse objectives, like the percentile criterion (also known as value-at-risk) (Delage and Mannor, 2010; Behzadian et al., 2021). In practice, robust MDP objectives tend to be overly conservative, thus soft-robustness (Derman et al., 2018) has been proposed as an alternative objective, which is identical to the risk-neutral objective of MMDPs.

In this chapter we approach uncertainty-aware optimization from a different perspective. Instead of fixing the policy optimization objective and designing a particular algorithm to solve for that objective, we propose a general-purpose method that aims to optimize *any* differentiable function of a learned distribution of values. The strength of our approach is that it flexibly accomodates an entire family of objectives that might suit different tasks, all within the same algorithm and with minimal changes.

3.2. Background & Notation

In this section, we provide the relevant background and formally introduce the problem of value distribution estimation. The problem statement and background share many similarities with Chapter 2, but we re-introduce the core concepts and notation for completeness.

We use upper-case letters to denote random variables and lower-case otherwise. The notation $\mathcal{P}(\mathcal{X})$ refers to the space of probability distributions over the set $\mathcal{X}$, such that $\nu \in \mathcal{P}(\mathcal{X})$ is a probability measure with the usual[1] σ-algebra. We forego measure-theoretic formalisms and further qualifications of measurability will be implied with respect to the usual σ-algebra (cf. Bellemare et al., 2023, Remark 2.1).

[1]Refers to the power set σ-algebra for finite $\mathcal{X}$, the Borel σ-algebra for infinite $\mathcal{X}$ and the product σ-algebra on products of such spaces (Bellemare et al., 2023).

3.2.1. Markov Decision Processes

We consider an agent that acts in an infinite-horizon MDP $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, p, r, \gamma\}$ with finite state space $\mathcal{S}$, finite action space $\mathcal{A}$, unknown transition function $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ that maps states and actions to the set of probability distributions over $\mathcal{S}$, a known[2] and bounded reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and a discount factor $\gamma \in [0, 1)$. The agent is equipped with an action-selection stochastic policy $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ that defines the conditional probability distribution $\pi(a \mid s)$, $(s, a) \in \mathcal{S} \times \mathcal{A}$. Given an initial state $s \in \mathcal{S}$ and a policy π, the RL agent interacts with the environment and generates a random *trajectory* $T = \{S_h, A_h, R_h\}_{h=0}^{\infty}$, with $S_0 = s$ and for $h \geq 0$ we have $A_h \sim \pi(\cdot \mid S_h)$, $R_h = r(S_h, A_h)$, $S_{h+1} \sim p(\cdot \mid S_h, A_h)$.

3.2.2. Return-Distributional Reinforcement Learning

The *return* of a policy, denoted Z^π, is a random variable defined as the discounted sum of rewards along a trajectory, $Z^\pi(s) = \sum_{h=0}^{\infty} \left[\gamma^h R_h \mid S_0 = s\right]$. The randomness in trajectories and returns originates from the inherent stochasticity of the environment dynamics and the policy, oftentimes called *aleatoric* uncertainty. A common objective for the RL agent is to maximize the *expected* return, where we average over this aleatoric noise to obtain a deterministic function known as the *value*. The value function of policy π under dynamics p, starting from $s \in \mathcal{S}$ is defined as a map $v^{\pi,p} : \mathcal{S} \to \mathbb{R}$ and is given by

$$v^{\pi,p}(s) = \mathbb{E}_T \left[\sum_{h=0}^{\infty} \gamma^h R_h \middle| S_0 = s, p \right], \tag{3.1}$$

where we explicitly condition on the dynamics p; although redundant in the standard RL setting, this notation will become convenient later when we consider a distribution over dynamics.

In contrast to learning the value function, *return-distributional* RL aims to learn the entire distribution of returns by leveraging the random variable *return-distributional* Bellman equation (Bellemare et al., 2017)

$$Z^\pi(s) \overset{D}{=} r(s, A) + \gamma Z^\pi(S'), \tag{3.2}$$

where $A \sim \pi(\cdot \mid s)$, $S' \sim p(\cdot \mid s, A)$ and $(\overset{D}{=})$ denotes equality in distribution, i.e., the random variables in both sides of the equation may have different outcomes, but they share the same probability distribution.

We adopt the same Bayesian framework described in Section 2.2.1, but the main focus of this chapter is to study the *value-distribution*[3] function $\mu^\pi : \mathcal{S} \to \mathcal{P}(\mathbb{R})$, such that $V^\pi(s) \sim \mu^\pi(s)$, $\forall s \in \mathcal{S}$. As such, μ^π represents the distribution of the *epistemic* noise around the *expected* return of a policy. In Figure 3.1, we illustrate in a simple MDP the fundamental difference between return distributions and value distributions: the former captures aleatoric noise from the decision process, while the latter models *epistemic* uncertainty stemming from uncertain MDP dynamics. Refer to Figure 3.2 for another example of an uncertain transition probability and the value distribution it induces. While both value and return distributions aim to obtain a richer representation of complex random variables, only the former characterizes the type of uncertainty that is valuable for effective exploration of the environment.

[2]The theory results can be easily extended to unknown reward functions.

[3]We focus on state-value functions for simplicity, but the results have a straightforward extension for state-action-value functions.

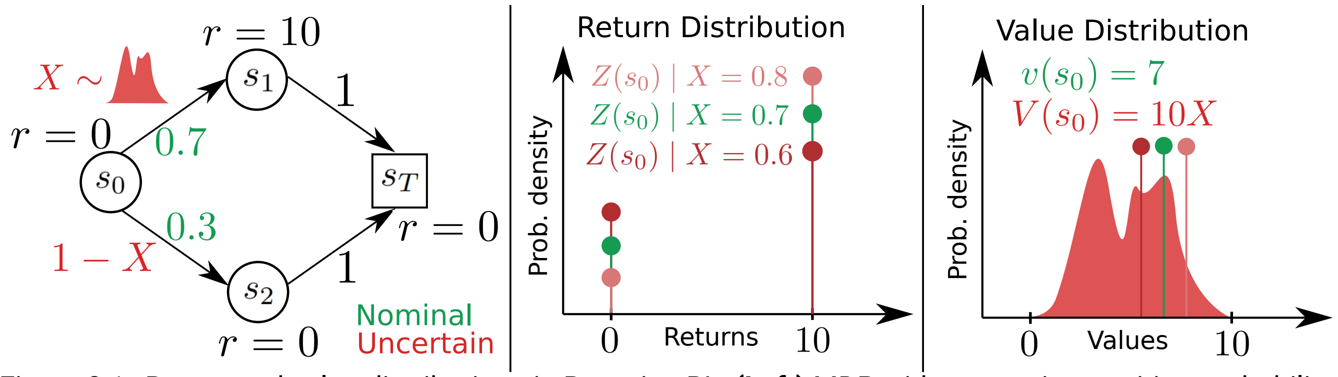


Figure 3.1.: Return and value distributions in Bayesian RL. **(Left)** MDP with uncertain transition probability from s_0 given by a random variable $X \in [0, 1]$. **(Middle)** Return distributions at s_0 for realizations of X, including the nominal dynamics (green). The return distribution captures the *aleatoric* noise under the sampled dynamics. **(Right)** Distribution of values at s_0. In the nominal case, the value $v(s_0)$ is a scalar obtained from averaging the aleatoric uncertainty of the return distribution $Z(s_0)$ under the nominal dynamics. In our setting, $V(s_0)$ is a random variable due to the *epistemic* uncertainty around the MDP dynamics. To sample from $V(s_0)$ is equivalent to first sample $X = \tilde{x}$, compute the *conditional* return distribution $Z(s_0)|X = \tilde{x}$ and finally average over the aleatoric noise.

3.3. The Value-Distributional Bellman Equation

In this section, we establish the theoretical backbone of iterative algorithms for learning the value-distribution function μ^π. We include formal proofs in Appendix B.1.

For the nominal transition kernel p, we can relate the values at subsequent time steps using the well-known Bellman equation

$$v^{\pi,p}(s) = \sum_a \pi(a \mid s) r(s,a) + \gamma \sum_{s',a} \pi(a \mid s) p(s' \mid s,a) v^\pi(s'), \tag{3.3}$$

which holds for any policy π and state $s \in \mathcal{S}$. The following statement is the straightforward extension of (3.3) in our Bayesian setting. While the result is not novel, it serves as a building block towards our main theoretical contribution.

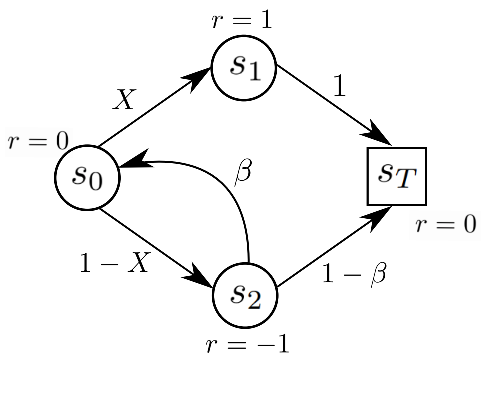
Proposition 1 (Random Variable Value-Distribution Bellman Equation). *Let V^π be the random value function defined in* (2.2). *Then, it holds that*

$$V^\pi(s) = \sum_a \pi(a \mid s) r(s,a) + \gamma \sum_{s',a} \pi(a \mid s) P(s' \mid s,a) V^\pi(s'), \tag{3.4}$$

for any policy π and initial state $s \in \mathcal{S}$.

Note that the random variable value-distribution Bellman equation in (3.4) differs from the random variable return-distribution Bellman equation in (3.2) in that the former holds in strict equality, while the latter holds in the weaker notion of equality in distribution. The main caveat of (3.4) with respect to model-free distributional RL is that, in general, $P(s' \mid s, a)$ and $V^\pi(s')$ are correlated.

We now shift from discussing the random value function to focus on the value distribution function. First, we provide a definition for μ^π that holds in the general case. Intuitively, we can think of μ^π as the result of



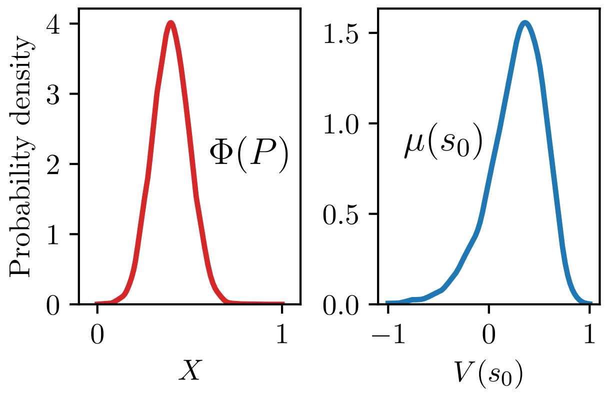


Figure 3.2.: Example value distribution. **(Left)** Uncertain MDP with a truncated Gaussian transition probability $X \sim \bar{\mathcal{N}}(\mu = 0.4, \sigma = 0.1)$ and a scalar (deterministic) $\beta \in [0, 1]$. For this example, we fixed $\beta = 0.9$. **(Middle)** Distribution over MDPs, which corresponds directly to the distribution of X. **(Right)** Corresponding distribution of values for state s_0.

pushing the probability mass of the posterior distribution Φ through the map defined by the value function (3.1). To formalize our statement, we leverage the notion of pushforward measures akin to Rowland et al. (2018).

Definition 1. Given measurable spaces $\mathcal{X}$ and $\mathcal{Y}$, a measurable function $f : \mathcal{X} \to \mathcal{Y}$ and a measure $\nu \in \mathcal{P}(\mathcal{X})$, the pushforward measure $f_{\#}\nu \in \mathcal{P}(\mathcal{Y})$ is defined by $f_{\#}\nu(B) = \nu(f^{-1}(B))$, for all Borel sets $B \subseteq \mathcal{Y}$.

Informally, given a random variable $X \sim \nu$ and the map f as in Definition 1, the pushforward of ν by f, denoted $f_{\#}\nu$, is defined as the distribution of the random variable $f(X)$ (Bellemare et al., 2023).

With slight abuse of notation, define the map $v^\pi : \mathcal{S} \times \mathscr{P} \to \mathbb{R}$ where $\mathscr{P}$ denotes the set of all transition functions[4] for $(\mathcal{S}, \mathcal{A})$, such that $v^\pi(s, p) = v^{\pi,p}(s)$ for any $p \in \mathscr{P}$, $s \in \mathcal{S}$. Then, the value distribution function μ^π is simply the pushforward measure of Φ by v^π.

Definition 2. The value distribution function is defined as

$$\mu^\pi = v^\pi_{\#}\Phi. \tag{3.5}$$

From Definition 2 we can already derive a simple (albeit, inefficient and computationally expensive) sample-based algorithm to estimate μ^π: sample from the posterior Φ and compute the value function (3.1) for each sample, which results in samples from μ^π. However, our main goal in this chapter is to find a recursive definition of μ^π such that we can introduce a simple, yet efficient estimation algorithm.

The main challenge in establishing a recursion for learning μ^π is the dependency between $P(s' \mid s, a)$ and $V^\pi(s')$ in (3.4). We side-step this issue by restricting our study to a family of MDPs under which these random variables are independent, as similarly done in Chapter 2. All the results that follow in this section hold under Assumptions 1–3. Despite these limitations, in Section 3.4 we empirically show that the algorithm stemming from our theory yields reasonable esimates of the value distribution μ^π in MDPs *with* cycles.

[4]The set of all transition functions can also be written as $\mathcal{P}(\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$ in standard set theory notation.

Beyond tabular representations of the transition function, introducing function approximation violates Assumption 1 due to generalization of the model (O'Donoghue et al., 2018; Zhou et al., 2020; Derman et al., 2020). However, in Section 3.7 our approach demonstrates strong empirical performance when paired with neural networks for function approximation.

We want to highlight that, under our assumptions, the *mean* value function $\mathbb{E}[V^\pi]$ corresponds exactly to the value function under the mean of the posterior Φ, denoted $\bar{p}$. That is, $\mathbb{E}[V^\pi] = v^{\pi,\bar{p}}$. If our ultimate goal is to estimate the mean of μ^π, then standard approaches to approximately solve the Bellman expectation equation suffice. However, in this chapter we motivate the need for the distributional approach as it allows to flexibly specify policy objectives *beyond* maximizing the mean of the values. For instance, in Section 3.7 we explore the performance of optimistic value objectives.

To establish a Bellman-style recursion defining the value distribution function, we use the notion of pushforward measure from Definition 1. In particular, we are interested in the pushforward of the value distribution by the bootstrap function in (3.4). First, for any MDP with transition function $p \in \mathscr{P}$, we denote by $p^\pi : \mathcal{S} \to \mathcal{P}(\mathcal{S})$ the transition function of the Markov Reward Process (MRP) *induced* by policy π, defined by $p^\pi(s' \mid s) = \sum_a \pi(a \mid s) p(s' \mid s, a)$. Further, it is convenient to adopt the matrix-vector notation of the standard Bellman equation: $\mathbf{v}^{\pi,p} = \mathbf{r}^\pi + \gamma \mathbf{p}^\pi \mathbf{v}^{\pi,p}$, where $\mathbf{r}^\pi \in \mathbb{R}^{\mathcal{S}}$, $\mathbf{v}^{\pi,p} \in \mathbb{R}^{\mathcal{S}}$ are vectors and $\mathbf{p}^\pi \in \mathbb{R}^{\mathcal{S}\times\mathcal{S}}_{[0,1]}$ is a so-called stochastic matrix whose entries are restricted to $[0,1]$ and whose rows sum up to 1, i.e., such that it represents the transition function p^π. Then, we define the bootstrap function $b_{r,p,\gamma} : \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$ applied to value vectors:

$$b_{r,p,\gamma} : \mathbf{v} \to \mathbf{r} + \gamma \mathbf{p} \mathbf{v}, \tag{3.6}$$

for an arbitrary $\mathbf{r} \in \mathbb{R}^{\mathcal{S}}$, $\mathbf{p} \in \mathbb{R}^{\mathcal{S}\times\mathcal{S}}_{[0,1]}$ and $\gamma \in [0,1)$. Applying $b_{r,p,\gamma}$ is a combination of adding $\mathbf{r}$ to a γ-scaled linear transformation of the input vector. Further, we express mixtures with weights given by the posterior $\Phi(P \mid \mathcal{D})$ more compactly with the notation[5] $\mathbb{E}_P[\cdot]$, where the argument is a probability distribution depending on P. Given the pushforward and mixture operations, we can now propose a Bellman equation for the value distribution function μ^π.

Lemma 1 (Value-Distribution Bellman Equation). *The value distribution function μ^π obeys the Bellman equation.*

$$\mu^\pi = \mathbb{E}_P\left[(b_{r^\pi,P^\pi,\gamma})_\# \mu^\pi\right] \tag{3.7}$$

for any policy π.

Lemma 1 provides the theoretical backbone towards designing an iterative algorithm for learning the value distribution function. In particular, the recursive definition for μ^π, which corresponds to a mixture of pushforwards of itself, leads to efficient estimation methods by dynamic programming. Alternatively, we can also write the value-distributional Bellman equation for each state $s \in \mathcal{S}$. With slight abuse of notation, define $b_{r,\gamma} : \mathbb{R} \to \mathbb{R}$ as the map $v \to r + \gamma v$, then

$$\mu^\pi(s) = \mathbb{E}_P\left[\sum_{s'} P^\pi(s' \mid s)(b_{r^\pi(s),\gamma})_\# \mu^\pi(s')\right]. \tag{3.8}$$

Note that (3.5) holds generally, while (3.7) and (3.8) only hold under Assumptions 1–3. Moreover, the operator $\mathbb{E}_P[\cdot]$ is well-defined in (3.7) and (3.8) since $P^\pi(s' \mid s)$ is bounded in $[0,1]$ for all $s, s' \in \mathcal{S}$ (Billingsley, 1995).

[5]Adapted from Bellemare et al. (2023). It refers to a mixture distribution and must not be mistaken by an expected value, which is a scalar.

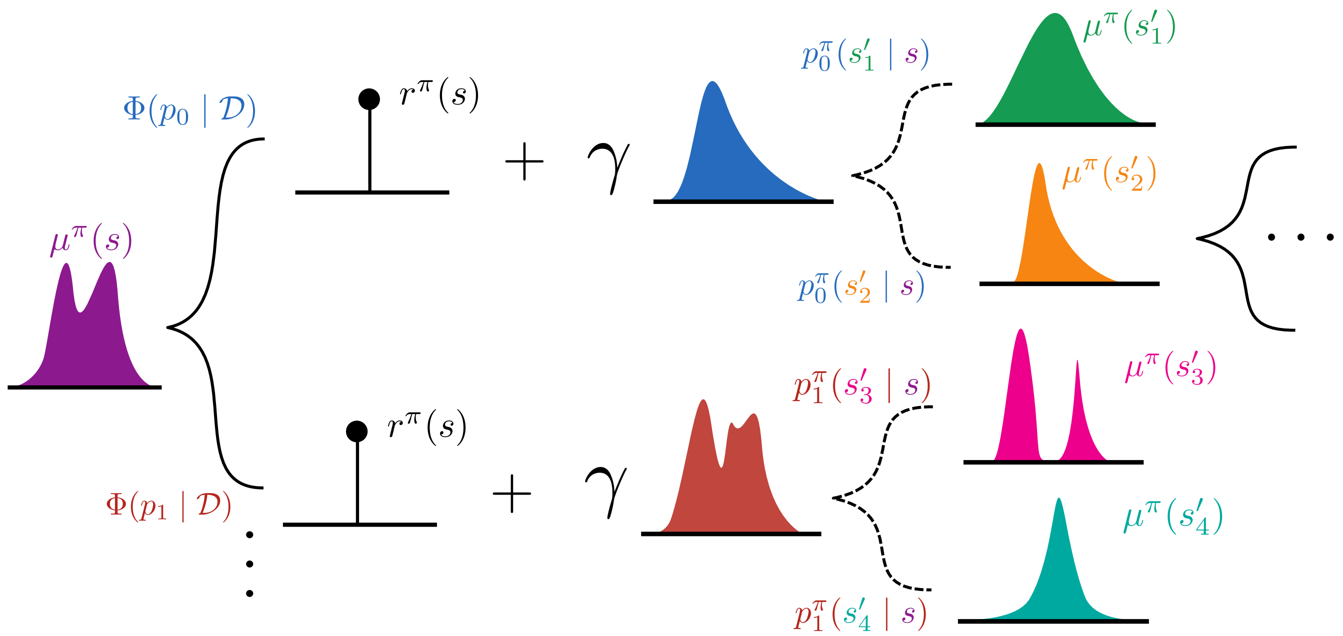


Figure 3.3.: Visualization of the value-distributional Bellman backups, as prescribed by (3.8). We identify four operations on distributions: infinite mixture over posterior transition functions (solid braces), shift by reward, scale by discount factor and mixture over next states (broken line braces)[6]. The main difference w.r.t the return-distributional backup (cf. Bellemare et al., 2023, Figure 2.6) is the presence of the two distinct mixture operations.

In Figure 3.3 we illustrate the core operations involved in the value-distributional Bellman recursion prescribed by (3.8).

From (3.7) we can extract an operator that acts on arbitrary value distribution functions.

Definition 3. The value-distributional Bellman operator $\mathcal{T}^\pi : \mathcal{P}(\mathbb{R})^\mathcal{S} \to \mathcal{P}(\mathbb{R})^\mathcal{S}$ is defined by

$$\mathcal{T}^\pi \mu = \mathbb{E}_P\left[(b_{r^\pi, P^\pi, \gamma})_\# \mu\right] \tag{3.9}$$

Intuitively, the operator $\mathcal{T}^\pi$ corresponds to mixing pushforward distributions, where the pushforward itself involves shifting, scaling and linearly transforming the probability mass. The natural question that follows is whether we can establish convergence to μ^π by repeated applications of $\mathcal{T}^\pi$ starting from an arbitrary initial guess μ_0.

Our convergence result is an adaptation of the standard distributional RL analysis by Bellemare et al. (2023). With some abuse of notation, we adopt the supremum p-Wasserstein distance $\bar{w}_p$ to establish contractivity of the operator $\mathcal{T}^\pi$ (see Definition 5 in Appendix B.1).

Theorem 3. *The operator $\mathcal{T}^\pi$ is a γ-contraction with respect to $\bar{w}_p$ for all $p \in [1, \infty)$. That is, $\bar{w}_p(\mathcal{T}^\pi \mu, \mathcal{T}^\pi \mu') \leq \gamma \bar{w}_p(\mu, \mu')$ for all $\mu, \mu' \in \mathcal{P}(\mathbb{R})^\mathcal{S}$ such that $V(s') \sim \mu(s')$, $V'(s') \sim \mu'(s')$ are conditionally independent of $P^\pi(s' \mid s)$ given $s' \in \mathcal{S}$.*

[6]The pushforward operator $(b_{r,\gamma})_\#$ is linear (cf. Bellemare et al., 2023, Exercise 2.13), so it can be moved outside the next-state mixture operation as depicted in the diagram.

Theorem 3 parallels similar results in standard RL and model-free distributional RL, in that it allows us to establish the convergence of iterated applications of $\mathcal{T}^\pi$ and characterize the operator's fixed-point.

Corollary 2. *Denote the space of value distribution functions with bounded support*[7] *by $\mathcal{P}_B(\mathbb{R})^\mathcal{S}$. Given an arbitrary value distribution function $\mu_0 \in \mathcal{P}_B(\mathbb{R})^\mathcal{S}$, the sequence $\{\mu_k\}_{k=0}^\infty$ defined by $\mu_{k+1} = \mathcal{T}^\pi \mu_k$ for all $k \geq 0$ is such that $\bar{w}_p(\mu_k, \mu^\pi) \leq \gamma^k \bar{w}_p(\mu_0, \mu^\pi) \to 0$ as $k \to \infty$ for $p \in [1, \infty)$. That is, μ^π is the unique fixed-point of the operator $\mathcal{T}^\pi$.*

Proof. We wish to apply Theorem 3 on the sequence of pairs $\{(\mu_k, \mu^\pi)\}_{k=0}^\infty$. The conditional independence assumption required to apply Theorem 3 holds for μ^π (see Lemma 3), but it is straightforward to show it also holds (under Assumptions 1–3) for all the elements of the sequence $\{\mu_k\}_{k=0}^\infty$ (see Lemma 9). Further, since we consider bounded rewards, it follows immediately that $\mu^\pi \in \mathcal{P}_B(\mathbb{R})^\mathcal{S}$. Moreover, it can be shown that the operator $\mathcal{T}^\pi$ maps $\mathcal{P}_B(\mathbb{R})^\mathcal{S}$ onto itself, such that for arbitrary $\mu \in \mathcal{P}_B(\mathbb{R})^\mathcal{S}$ then $\mathcal{T}^\pi \mu \in \mathcal{P}_B(\mathbb{R})^\mathcal{S}$ (see Lemma 10). By Theorem 3, $\mathcal{T}^\pi$ is then a *contraction mapping* and by Banach's fixed-point theorem $\mathcal{T}^\pi$ admits a unique fixed-point which is the limiting value of the sequence $\{\mu_k\}_{k=0}^\infty$. Since $\mu^\pi = \mathcal{T}^\pi \mu^\pi$ holds by Lemma 1, then μ^π must be the unique fixed-point of $\mathcal{T}^\pi$. □

In summary, Corollary 2 establishes that repeated applications of $\mathcal{T}^\pi$ from an arbitrary initial guess converges to the value distribution function μ^π. Inspired by this theoretical result, in the remaining sections we introduce and evaluate a practical algorithm for learning the value distribution function.

3.4. Quantile-Regression for Value-Distribution Learning

In the previous section we described an iterative process that converges to μ^π starting from an arbitrary value distribution with bounded support. In practice, however, to implement such a recursion we must project the value distributions onto some finite-dimensional parameter space. Following the success of quantile distributional RL (Dabney et al., 2018b), we adopt the quantile parameterization. Let $\mathcal{V}_m$ be the space of quantile distributions with m quantiles and corresponding quantile levels $\tau_i = 1/m$ for $i = \{1, \dots, m\}$ and $\tau_0 = 0$. Define a parametric model $q : \mathcal{S} \to \mathbb{R}^m$, then the quantile distribution $\mu_q \in \mathcal{V}_m$ maps states to a uniform probability distribution supported on $q_i(s)$. That is, $\mu_q(s) = \frac{1}{m}\sum_{i=1}^m \delta_{q_i(s)}$, where δ_x denotes the Dirac delta distribution centered at $x \in \mathbb{R}$, such that $\mu_q(s)$ is a uniform mixture of Dirac deltas where the particle $q_i(s)$ corresponds to the τ_i-quantile at state s. With this parameterization, our aim now becomes to compute the so-called quantile projection of μ^π onto $\mathcal{V}_m$, given by

$$\Pi_{w_1} \mu^\pi := \underset{\mu_q \in \mathcal{V}_m}{\operatorname{argmin}}\, w_1(\mu^\pi, \mu_q), \tag{3.10}$$

where w_1 is the 1-Wasserstein distance. Define $F_{\mu^\pi}^{-1}$ as the inverse cumulative distribution function of μ^π, then the distance metric becomes

$$w_1(\mu^\pi, \mu_q) = \sum_{i=1}^m \int_{\tau_{i-1}}^{\tau_i} \left| F_{\mu^\pi}^{-1}(\omega) - q_i \right| d\omega, \tag{3.11}$$

[7]Under bounded reward functions, the corresponding value distributions have bounded support. The corollary can be relaxed to distributions with bounded moments (see Proposition 4.30 in Bellemare et al. (2023).)

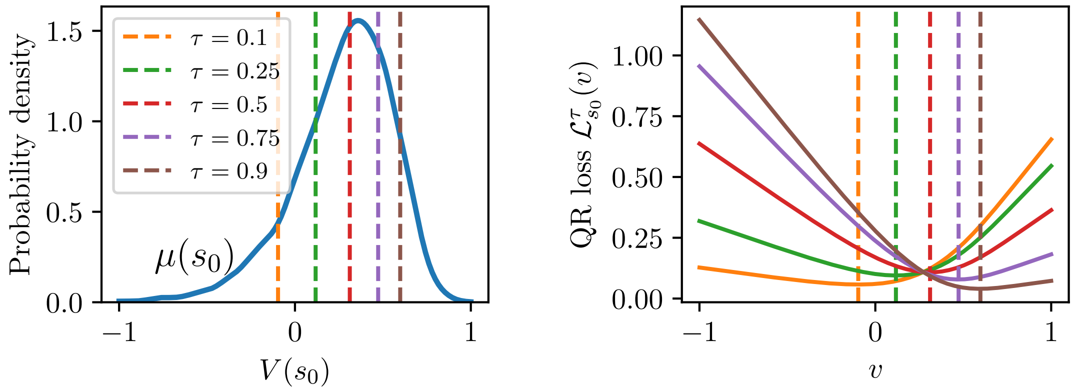


Figure 3.4.: Quantile-regression loss for the example MDP of Figure 3.2. **(Left)** Probability density of values for state s_0, with five quantile levels in colored vertical lines. **(Right)** The quantile regression loss (3.12) for the five quantile levels; the vertical lines correspond to the minimum of the color-matching loss. The vertical lines on both plots match upto numerical precision, meaning that following the gradient of such a convex loss function would indeed converge to the quantile projection $\Pi_{w_1}\mu$.

since μ_q is a uniform distribution over m Dirac deltas with support $\{q_1, \dots, q_m\}$. Let $\hat{\tau}_i = (2i-1)/2m$, then a valid minimizer of (3.10) exists and is achieved by selecting $q_i = F^{-1}_{\mu^\pi}(\hat{\tau}_i)$ (cf. Dabney et al., 2018b, Lemma 2). In summary, quantile projection as defined in (3.10) is equivalent to estimating each $\hat{\tau}_i$-quantile of μ^π.

We follow closely the treatment by Rowland et al. (2023) of quantile-regression temporal-difference learning for return-distributions and adapt it to instead work on value-distributions. The following loss function corresponds to the quantile-regression problem of estimating the τ-quantile of the value distribution μ^π:

$$\mathcal{L}^{\tau,\pi}_{s}(v) = \mathbb{E}_P\Big[\big(\tau\mathbb{1}\{V^\pi(s) > v\} + (1-\tau)\mathbb{1}\{V^\pi(s) < v\}\big)\big|V^\pi(s) - v\big|\Big]. \tag{3.12}$$

It is an asymmetric convex loss function, where quantile overestimation and underestimation errors are weighted by τ and $1-\tau$, respectively. The unique minimizer of this loss is the τ-quantile of μ^π, which we illustrate with an example in Figure 3.4.

Our goal is to propose a practical algorithm to learn the value distribution function based on the quantile-regression loss (3.12). If we have access to samples of $V^\pi(s)$, denoted $\tilde{v}^\pi(s)$, then we can derive an unbiased estimate of the negative gradient of (3.12) and obtain the update rule

$$q_i(s) \leftarrow q_i(s) + \alpha\big(\tau_i - \mathbb{1}\{\tilde{v}^\pi(s) < q_i(s)\}\big), \tag{3.13}$$

where α is some scalar step size. One option to sample $V^\pi = \tilde{v}^\pi$ would be to first sample a model $P = \tilde{p}$ and then solve the corresponding Bellman equation. Instead, we use a computationally cheaper alternative (albeit biased) and bootstrap like in temporal-difference learning, so that the samples are defined as

$$\tilde{v}^\pi(s) = r^\pi(s) + \gamma \sum_{s'} \tilde{p}^\pi(s' \mid s) q_J(s'), \tag{3.14}$$

where $J \sim \text{Uniform}(1, \dots, m)$. Lastly, we reduce the variance of the gradient estimate by averaging over the

Algorithm 4 Epistemic Quantile-Regression (EQR)

```
1: **Input:** Posterior MDP $\Phi$, policy $\pi$, number of quantiles $m$.
2: Randomly initialize estimates $\{q_i(s)\}_{i=1}^m$ for all $s \in \mathcal{S}$
3: **repeat**
4:     Sample $\tilde{p}$ from posterior $\Phi$
5:     **for** $i = 1, \dots, m$ **do**
6:         Update $q_i(s)$ with (3.15) for all $s \in \mathcal{S}$
7: **until** convergence
8: **return** $\{q_i(s)\}_{i=1}^m$
```

values of J, which results in the update

$$q_i(s) \leftarrow q_i(s) + \frac{\alpha}{m}\left(\tau_i - \sum_{j=1}^{m} \mathbb{1}\left\{ r^\pi(s) + \gamma \sum_{s'} \tilde{p}^\pi(s' \mid s) q_j(s') < q_i(s) \right\}\right). \tag{3.15}$$

We introduce EQR in Algorithm 4 to iteratively learn the value distribution function μ^π. From an arbitrary initial guess of quantiles, we sample an MDP from the posterior and update the quantiles using (3.15) for all states until convergence. The following examples illustrate the performance of EQR in tabular problems.

Example 1 (Toy MDP). *Consider once more the tabular MDP of Figure 3.2. Our goal is to assess the convergence properties of EQR both when Assumptions 1–3 hold, but also when they are violated* [8]*. We control the degree of violation of Assumptions 2 and 3 via the parameter β of the MDP. If $\beta = 0$, the assumptions hold and $V(s_0)$ and $P(s_2|s_0)$ are decorrelated. As β goes from zero to one, the covariance between these two random variables increases monotonically. We manually design three MDP posterior distributions that result in diverse distributions for $V(s_0)$. The value distributions shown in the top row of Figure 3.5 are the result of modelling the MDP parameter X as the following mixtures of truncated Gaussian distributions: $X_{\text{left}} \sim \bar{\mathcal{N}}(\mu = 0.5, \sigma = 0.1)$, $X_{\text{middle}} \sim 0.5\bar{\mathcal{N}}(\mu = 0.3, \sigma = 0.03) + 0.5\bar{\mathcal{N}}(\mu = 0.6, \sigma = 0.05)$ and $X_{\text{right}} \sim 0.5\bar{\mathcal{N}}(\mu = 0.3, \sigma = 0.03) + 0.5\bar{\mathcal{N}}(\mu = 0.5, \sigma = 0.15)$. For this selection of value distributions, we run EQR to estimate $m = 10$ quantiles.*

The middle row of Figure 3.5 shows that, for $\beta = 0$, the prediction error oscillates close to zero for every quantile, thus validating the result of Corollary 2. To test the prediction quality when the assumptions are violated, we generate different values for $\beta \in [0, 1]$ and run EQR for the same three MDP posterior distributions. The bottom plots in Figure 3.5 show the 1-Wasserstein metric between the true and predicted quantile distributions after 10^4 gradient steps; the error, like the covariance between $V(s_0)$ and $P(s_2|s_0)$, increases monotonically with β. The prediction quality thus degrades depending on the magnitude of the covariance between the transition kernel and the values.

Example 1 is mostly pedagogical and serves the purpose of validating our theoretical result, but it remains a contrived example with limited scope. The next example analyzes the performance of EQR in a standard tabular problem.

[8]In order to be closer to standard settings, when the assumptions are violated (i.e., MDP contains cycles) we do not unroll the MDP as described in Figure 2.1.

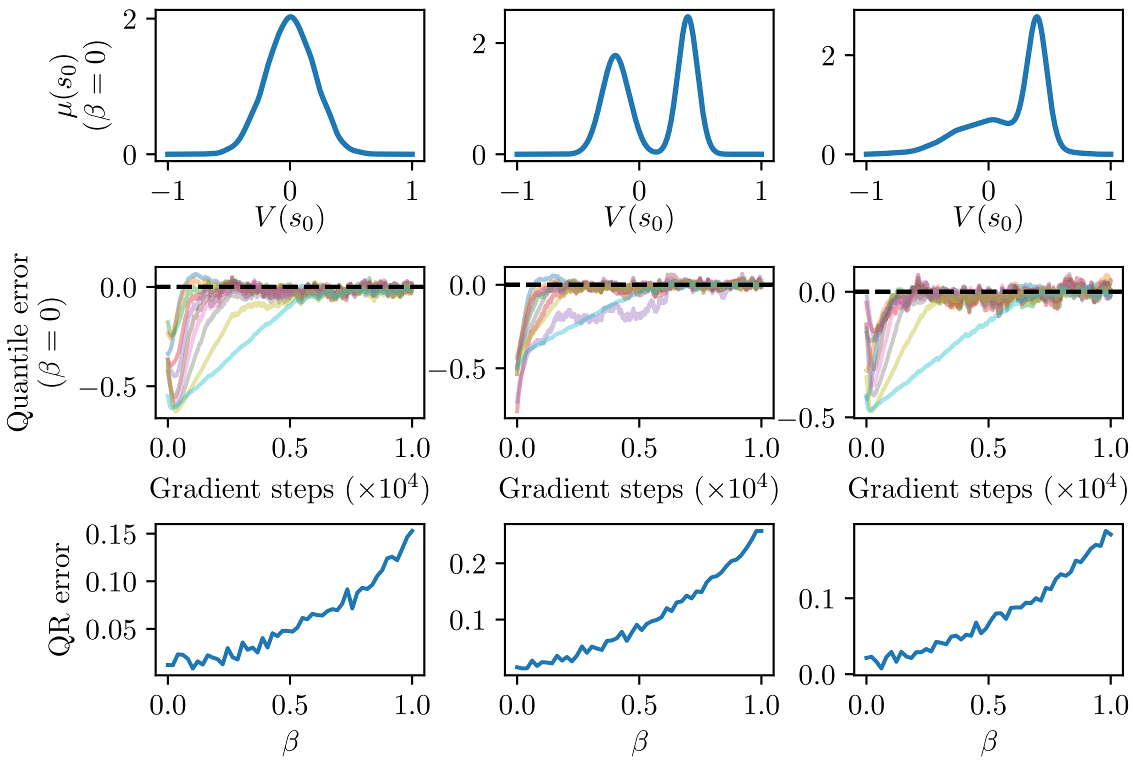


Figure 3.5.: Performance of quantile-regression for value-distribution learning in the example MDP of Figure 3.2. The parameter β controls the covariance between $V(s_0)$ and $P(s_2|s_0)$; the covariance increases with β and is zero for $\beta = 0$. **(Top)** Value distributions (Gaussian, bi-modal and heavy-tailed) generated by different prior distributions of the parameter δ. **(Middle)** Evolution of the per-quantile estimation error $(\Pi_{w_1}\mu(s_0) - \mu_q(s_0))$ between the true quantile projection and the prediction; for $\beta = 0$, our algorithm oscillates around the true quantile projection. **(Bottom)** 1-Wasserstein metric between the true quantile projection and the estimate μ_q after 10^4 gradient steps, as a function of the correlation parameter β. As β moves from zero to one, the regression error increases and the algorithm no longer converges to the true quantiles, although the error is relatively small.

Example 2 (Gridworld)**.** *We consider a modification of the N-room gridworld environment by Domingues et al. (2021), consisting of three connected rooms of size 5×5. The task for this example is to predict $m = 100$ quantiles of the value distribution under the optimal policy $\pi^\star$ (obtained by running any standard tabular exploration algorithm, like PSRL (Osband et al., 2013)). We use a Dirichlet prior for the transition kernel and a standard Gaussian for the rewards. We collect data using $\pi^\star$, update the posterior MDP and run EQR to predict the value distribution.*

In Figure 3.6, we summarize the results at three different points during data collection. As more data is collected, the corresponding MDP posterior shrinks and we observe the value distribution concentrates around the value under the true dynamics (dotted vertical line). For both wide (episode 1) and narrow (episode 100) posteriors, EQR is able to accurately predict the distribution of values. The impact of violating Assumption 2 is the non-zero steady-state quantile-regression error. We observe the bias of the predicted quantiles is typically lowest (near zero) close to the median and highest at the extrema.

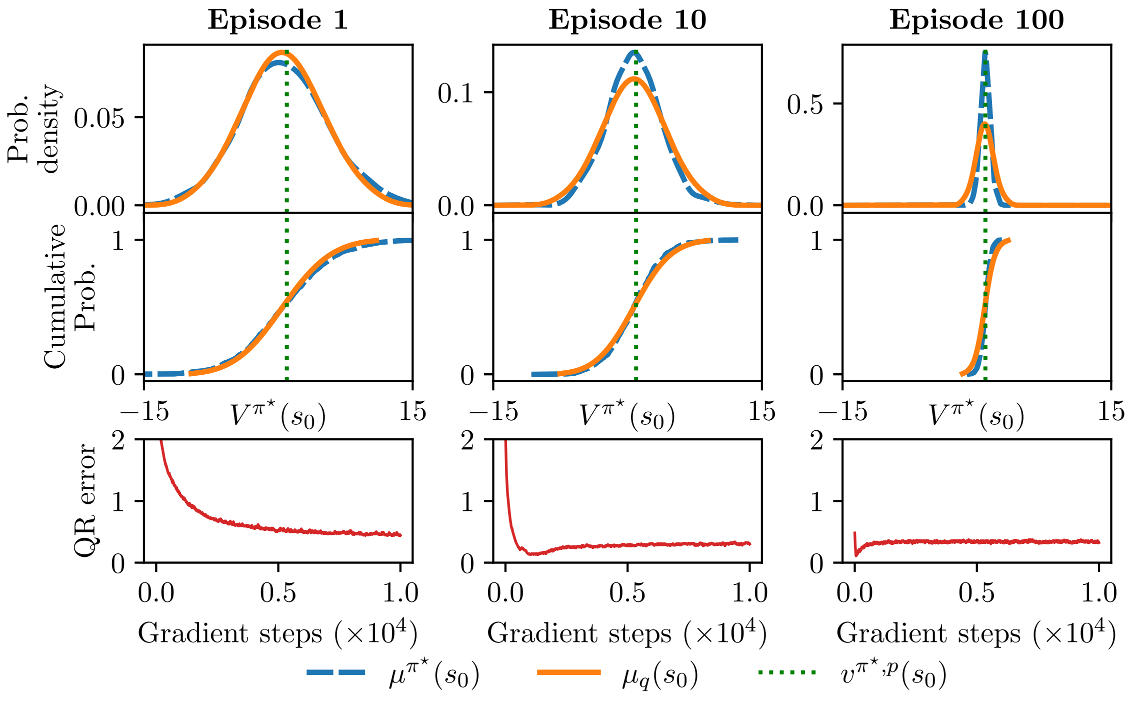


Figure 3.6.: Performance of EQR in a Gridworld environment. We train the optimal policy $\pi^\star$ using PSRL (Osband et al., 2013) and then use it for data collection. At different points during data collection, we run EQR to estimate $m = 100$ quantiles of the value distribution for the initial state under $\pi^\star$, given the current posterior MDP. **(Top-Middle)** PDF and CDF of the true (dashed blue) and predicted (solid orange) value distribution, with the true optimal value (dotted green) as vertical reference. **(Bottom)** 1-Wasserstein distance between the true quantile projection and the prediction.

3.5. Policy Optimization with Value Distributions

In this section, we propose an algorithm to optimize a policy given some differentiable utility function f of the learned quantile distribution μ_q (which is implicitly parameterized by π). Namely, we define the optimal policy

$$\pi^\star = \operatorname*{argmax}_{\pi} f(\mu_q; \pi). \tag{3.16}$$

To approximately solve (3.16), we combine EQR with SAC (EQR-SAC) to obtain a model-based reinforcement learning algorithm that leverages a value-distributional critic for policy optimization. Our algorithm is agnostic to f as long as it is differentiable and thus can backpropagate gradients through it. The key ingredients of our method are: (1) an ensemble-based posterior over MDPs, (2) a quantile-distributional critic network that models the m-quantile function $q(s, a)$ and (3) a policy network π_ϕ trained to optimize (3.16).

Posterior Dynamics. We adopt the baseline architecture from MBPO (Janner et al., 2019) and the implementation from Pineda et al. (2021), where the posterior MDP, denoted Γ_ψ, is represented as an ensemble of n neural networks trained via supervised learning on the environment dataset $\mathcal{D}$ to predict the mean and variance of a Gaussian distribution over next states and rewards. We use Γ_ψ to populate an experience replay buffer $\mathcal{D}_{\text{model}}$ with model-consistent k-step rollouts; that is, we use a consistent ensemble member throughout a rollout, rather than randomizing the model per-step like in MBPO.

Quantile Huber loss. We adopt the quantile Huber loss from Dabney et al. (2018b) in order to train the distributional critic. The Huber loss is given by

$$\mathcal{L}_\kappa(u) = \begin{cases} \frac{1}{2}u^2, & \text{if } |u| \leq \kappa \\ \kappa(|u| - \frac{1}{2}\kappa) & \text{otherwise} \end{cases}, \tag{3.17}$$

and the quantile Huber loss is defined by

$$\rho_\tau^\kappa(u) = \big|\tau - \delta(u < 0)\big|\mathcal{L}_\kappa(u). \tag{3.18}$$

For $\kappa = 0$, we recover the standard quantile regression loss, which is not smooth as $u \to 0$. In all our experiments we fix $\kappa = 1$ and to simplify notation define $\rho_\tau^1 = \rho_\tau$.

Critic. We train the critic on mini-batches drawn from $\mathcal{D}_{\text{model}}$, use the entropy regularization loss from SAC with temperature α and replace the quantile regression loss with the quantile Huber loss $\rho_\tau(u)$

$$\mathcal{L}_{\text{critic}}(q) = \mathbb{E}_{(S,A)\sim\mathcal{D}_{\text{model}}}\left[\mathbb{E}_{(\hat{R},\hat{P})\sim\Gamma_\psi}\left[\sum_{i=1}^{m}\mathbb{E}_J\left[\rho_{\tau_i}\left(\mathcal{T}q_J(S,A) - q_i(S,A)\right)\right]\right]\right], \tag{3.19}$$

where the target quantiles $\mathcal{T}q_j$ are defined as

$$\mathcal{T}q_j(s,a) = \hat{R}(s,a) + \gamma\,\mathbb{E}_{(S',A')\sim\hat{P}(\cdot|s,a),\pi_\phi}\big[q_j(S',A') - \alpha\log\pi_\phi(A' \mid S')\big]. \tag{3.20}$$

The expectation in (3.20) is approximated by generating transition tuples (s', a') using the policy and the sampled dynamics from Γ_ψ. Typically, model-based algorithms like MBPO only use data in the mini-batch to compose critic targets, rather than leveraging the learned dynamics model for better approximation of expectations.

Actor. The policy is trained to maximize the objective in (3.16), in addition to the entropy regularization term from SAC,

$$\mathcal{L}_{\text{actor}}(\phi) = \mathbb{E}_{S\sim\mathcal{D}_{\text{model}}}\Big[\mathbb{E}_{A\sim\pi_\phi}\big[f(q(S,A)) - \alpha\log\pi_\phi(A \mid S)\big]\Big]. \tag{3.21}$$

Let $\bar{q}(s,a)$ and $\sigma_q(s,a)$ be the mean and standard deviations of the quantiles, respectively. Then, we consider two concrete utility functions: the classical mean objective $f_{\text{mean}}(q(s,a)) = \bar{q}(s,a)$ and an objective based on optimism in the face of uncertainty $f_{\text{ofu}} = \bar{q}(s,a) + \sigma_q(s,a)$.

3.6. EQR-SAC Algorithm

A detailed execution flow for training an EQR-SAC agent is presented in Algorithm 5. Further implementation details are now provided.

Model learning. We use the `mbrl-lib` Python library from Pineda et al. (2021) to train N neural networks (Line 7). Our default architecture consists of four fully-connected layers with 200 neurons each (for the Quadruped environments we use 400 neurons to accomodate the larger state space). The networks predict delta states, $(s' - s)$, and receives as inputs normalizes state-action pairs. The normalization statistics

Algorithm 5 Epistemic Quantile-Regression with Soft Actor-Critic (EQR-SAC)

```
1:  Initialize policy $\pi_\phi$, MDP ensemble $\Gamma_\psi$, quantile critic $q$, environment dataset $\mathcal{D}$, model dataset $\mathcal{D}_{\text{model}}$,
    utility function $f$.
2:  Warm-up $\mathcal{D}$ with rollouts under $\pi_\phi$
3:  global step $\leftarrow 0$
4:  for episode $t = 0, \dots, T-1$ do
5:    for $E$ steps do
6:      if global step % $F == 0$ then
7:        Train $\Gamma_\psi$ on $\mathcal{D}$ via maximum likelihood
8:        for each MDP dynamics in $\Gamma_\psi$ do
9:          for $L$ model rollouts do
10:           Perform $k$-step rollouts starting from $s \sim \mathcal{D}$; add to $\mathcal{D}_{\text{model}}$
11:     Take action in environment according to $\pi_\phi$; add to $\mathcal{D}$
12:     for $G$ gradient updates do
13:       Update $\{q_i\}_{i=1}^m$ with mini-batches from $\mathcal{D}_{\text{model}}$, via SGD on (3.19)
14:       Update $\pi_\phi$ with mini-batches from $\mathcal{D}_{\text{model}}$, via SGD on (3.21)
15:   global step $\leftarrow$ global step $+1$
```

are updated each time we train the model and are based on the training dataset $\mathcal{D}$. We use the default initialization that samples weights from a truncated Gaussian distribution, but we increase by a factor of 2 the standard deviation of the sampling distribution.

Capacity of $\mathcal{D}_{\text{model}}$. The capacity of the model buffer is computed as $k \times L \times F \times N \times \Delta$, where Δ is the number of model updates we want to retain data in the buffer. That is, the buffer is filled only with data from the latest Δ rounds of model training and data collection (Lines 6-10).

Critic Loss. The distributional critic is updated in Line 13, for which we use the loss function (3.19). To approximate the target quantiles (3.20), we use the learned generative model and the policy to generate transition tuples (r, s', a'). More specifically, each (s, a) pair in a mini-batch from $\mathcal{D}_{\text{model}}$ is repeated X times and passed through every member of the ensemble of dynamics, thus generating n batches of X predictions (r, s'). Then, every s' prediction is repeated Y times and passed through π_ϕ, thus obtaining XY next state-action pairs (s', a'). This generated data is finally used in (3.20) to better approximate the expectation. In our experiments we use $X = Y$ and keep their product as a hyperparameter controlling the total amount of samples we use to approximate the expectation.

3.7. Experiments

In this section, we evaluate EQR-SAC in environments with continuous state-action spaces. We use a single codebase for all experiments and share architecture components amongst baselines whenever possible. The execution of experiments follows the workflow of Algorithm 5. The SAC base implementation follows the open-source repository `https://github.com/pranz24/pytorch-soft-actor-critic` and we allow for either model-free or model-based data buffers for the agent's updates, as done in `mbrl-lib`. We include specific hyperparameters used in experiments in Appendix B.2. Unless noted otherwise, all training curves are smoothened by a moving average filter and we report the mean and standard error over 10 random seeds.

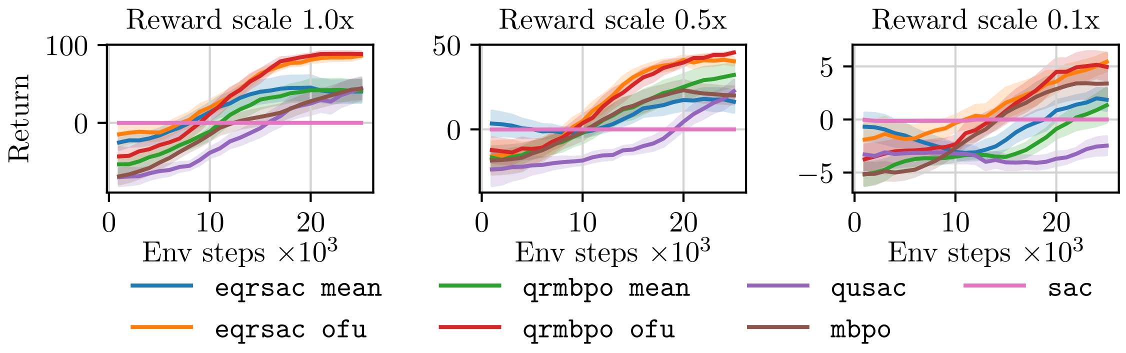


Figure 3.7.: Performance in the Mountain Car environment. We consider the original version of the environment (left) and two variants (middle, right) that scale down the rewards by some factor (0.5 and 0.1, respectively).

3.7.1. Baselines

SAC with typical design choices like target networks (Mnih et al., 2013), clipped double Q-learning (Fujimoto et al., 2018) and automatic entropy tuning (Haarnoja et al., 2019).

MBPO with slight modifications from Janner et al. (2019): (1) it only uses $\mathcal{D}_{\text{model}}$ to update the actor and critic, rather than mixing in data from $\mathcal{D}$; (2) it uses a fixed rollout length k, instead of an adaptive scheme. With respect to EQR-SAC, MBPO collects data differently: instead of collecting k-step rollouts under each model of the ensemble, it does so by uniformly sampling a new model per step of the rollout.

QR-MBPO, which replaces the critic in MBPO with a quantile-distributional critic, trained on the standard quantile-regression loss from Wurman et al. (2022), but using data from $\mathcal{D}_{\text{model}}$,

$$\mathcal{L}_{\text{critic}}^{\text{qrmbpo}}(q) = \mathbb{E}_{(S,A,S',R)\sim\mathcal{D}_{\text{model}}}\left[\left[\sum_{i=1}^{M}\mathbb{E}_J\left[\rho_{\tau_i}\left(\mathcal{T}q_J^{\text{qrmbpo}}(R,S,A) - q_i(S,A)\right)\right]\right]\right], \tag{3.22}$$

where the target quantile is defined as

$$\mathcal{T}q_j^{\text{qrmbpo}}(r,s',a) = r + \gamma\big(q_j(s',a') - \alpha\log\pi_\phi(a' \mid s')\big), \tag{3.23}$$

and $a' \sim \pi_\phi(\cdot \mid s')$. Importantly, (3.22) differs from (3.19) in that the former captures both the aleatoric and epistemic uncertainty present in $\mathcal{D}_{\text{model}}$, while the latter aims to average out the aleatoric noise from the target quantiles. The objective function for the actor is the same as EQR-SAC.

QU-SAC, as proposed in Chapter 2. It collects data as in EQR-SAC, but stores the n model-consistent rollouts in n separate buffers (while EQR-SAC uses a single buffer). Then it trains an ensemble of n standard critics on the corresponding n model-buffers. As such, it interprets the ensemble of critics as samples from the value distribution. The actor is optimized to maximize the mean prediction of the critic ensemble.

3.7.2. Case Study - Mountain Car

We motivate the importance of learning a distribution of values with a simple environment, the Mountain Car (Sutton and Barto, 2018) with continuous action space, as implemented in the `gym` library (Brockman

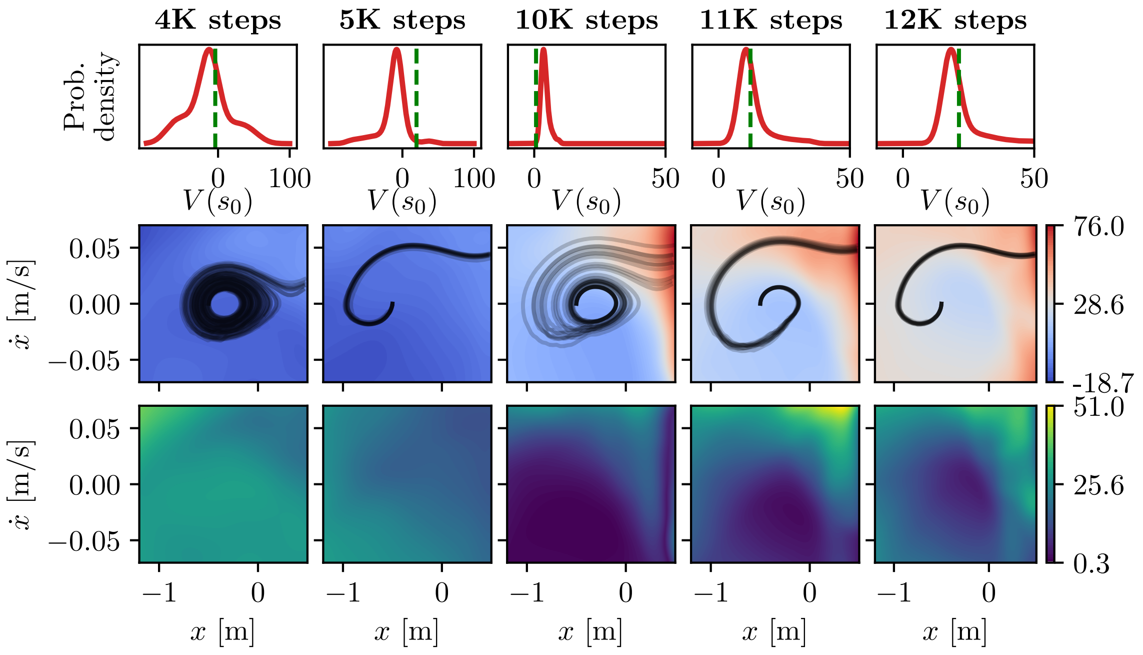


Figure 3.8.: Visualization of the learned value distribution of EQR-SAC at different points during training in the Mountain Car environment (with reward scale of 0.5x). **(Top)** The predicted value distribution at the initial state. The dotted line is the empirical value of s_0 based on ten trajectories (black lines of middle row). **(Mid-Bottom)** The mean (mid) and standard deviation (bottom) of the value distribution across the state space.

et al., 2016). The environment's rewards are composed of a small action penalty and a large sparse reward once the car goes up the mountain, defined by a horizontal position $x > 0.45$ meters. We consider three versions of the problem: the original one, and two variants where all the rewards are scaled by a constant factor of 0.5 and 0.1, respectively.

While it is low-dimensional and has simple dynamics, many RL algorithms fail to solve the Mountain Car problem due to its combination of action penalties and sparse rewards. Naive exploration strategies based on injecting unstructured noise, like SAC, typically fail to solve such tasks (Raffin et al., 2021). We plot the performance of EQR-SAC and all baselines in Figure 3.7. EQR-SAC and QR-MBPO have the best overall performance, both using the optimistic objective function f_{ofu} (as previously defined after (3.21)). These results highlight the need to model uncertainty *and* leverage it during optimization; optimizing the mean values significantly degraded performance of the distributional approaches.

In Figure 3.8, we inspect further the distribution of values learned by EQR-SAC during a training run. The value distribution is initially wide and heavy-tailed, as the agent rarely visits goal states. At 5K steps, the policy is close-to-optimal but the predicted distribution underestimates the true values. In subsequent steps, the algorithm explores other policies while reducing uncertainty and calibrating the predicted value distribution. At 12K steps, the algorithm stabilizes again at the optimized policy, but with a calibrated value distribution whose mean is close to the empirical value. We notice the large uncertainty in the top-right corner of the state space remains (and typically does not vanish if we run the algorithm for longer); we hypothesize this is mainly an artifact of the discontinuous reward function, which is smoothened out differently by each member of the ensemble of dynamics, such that epistemic uncertainty stays high.

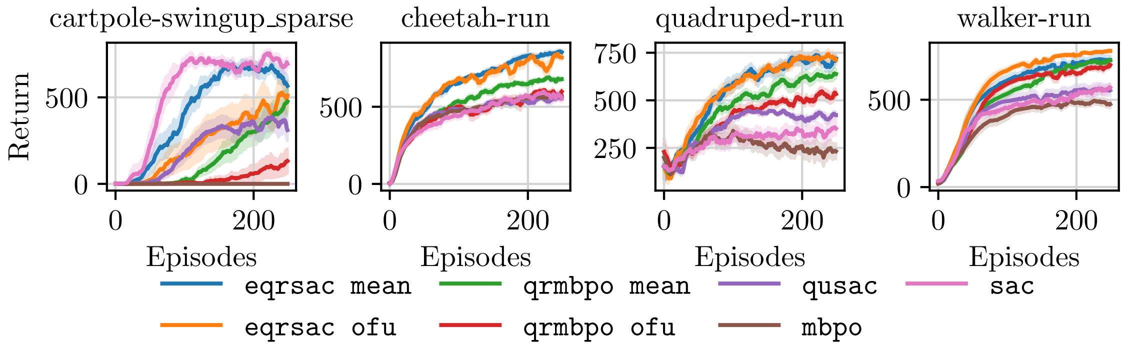


Figure 3.9.: Performance in four DeepMind Control tasks. Cartpole swing-up has sparse rewards, while Cheetah, Quadruped and Walker have dense rewards. EQR-SAC significantly improves performance with respect to the model-based baselines.

3.7.3. DM Control Benchmark

In order to evaluate EQR-SAC more broadly, we conduct an experiment in a subset of 16 continuous-control tasks from the DeepMind Control Suite (Tunyasuvunakool et al., 2020). The chosen environments include both dense/sparse rewards and smooth/discontinuous dynamics. In Figure 3.9, we plot the results for four environments ranging from small (cartpole) to mid/large (quadruped) observation spaces. Our method significantly improves performance over previous model-based algorithms in these environments. Moreover, in the full benchmark, EQR-SAC achieves the best (or comparable) final performance in 13 out of 16 tasks (see Appendix B.4). We summarize the results of the DMC benchmark in Figure 3.10 and Table 3.1, following the guidelines by Agarwal et al. (2021). At 250 episodes of training, EQR-SAC OFU achieves the highest normalized IQM score, which is $\sim 17\%$ higher than QR-MBPO mean. However, there exists some overlap between the 95% confidence intervals, which tend to be large in our benchmark due to a wide range of normalized scores across different environments. In this scenario, the recommendation in Agarwal et al. (2021) is to analyze score distribution performance profiles, as presented in Figure 3.10, which provide a more complete overview of the results. We observe the EQR-SAC OFU performance profile tends to dominate over the baselines, especially for normalized scores between $[0.5, 0.8]$.

Table 3.1.: Normalized inter-quartile mean scores in DMC benchmark after 100 and 250 episodes. For each aggregation metric, we report the point estimate and the 95% bootstrap confidence interval within parentheses, following the methodology in Agarwal et al. (2021). We **bold** the highest mean scores in each case.

Method	IQM-100	IQM-250
● `eqrsac mean`	**0.63** (0.55, 0.72)	0.73 (0.65, 0.81)
● `eqrsac ofu`	0.61 (0.53, 0.69)	**0.76** (0.69, 0.81)
● `qrmbpo mean`	0.46 (0.37, 0.56)	0.65 (0.56, 0.73)
● `qrmbpo ofu`	0.46 (0.38, 0.55)	0.64 (0.56, 0.69)
● `qusac`	0.41 (0.33, 0.50)	0.51 (0.43, 0.58)
● `mbpo`	0.30 (0.23, 0.39)	0.32 (0.24, 0.42)
● `sac`	0.54 (0.46, 0.61)	0.59 (0.52, 0.66)

We observe a clear gap in performance between MBPO and QR-MBPO, which supports the observations from Lyle et al. (2019) and reinforces their hypothesis that distributional critics boost performance under

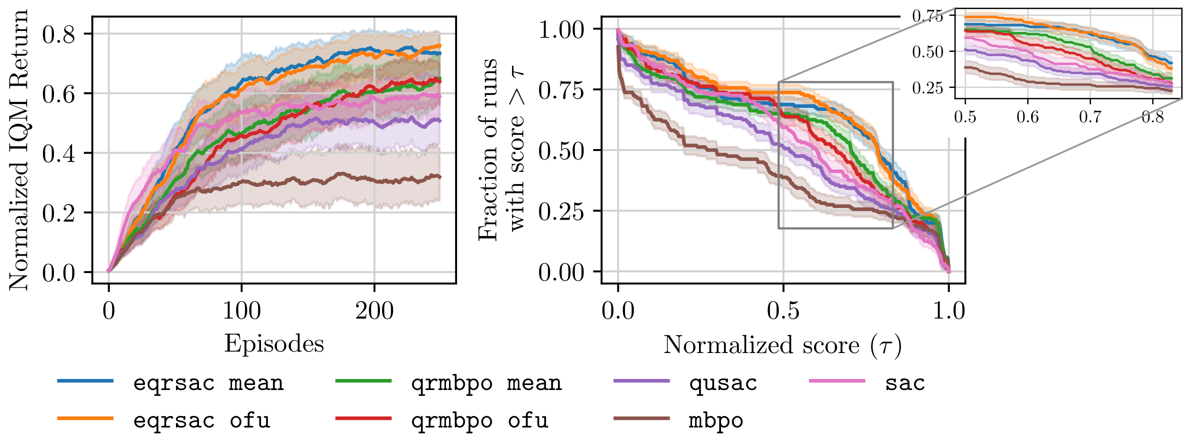


Figure 3.10.: Aggregated performance in DMC benchmark with 95% bootstrap confidence intervals. **(Left)** Inter-quartile mean returns normalized by the maximum achievable score of 1000. **(Right)** Performance profile at 250 episodes of training, with zoom in region with the most spread in results. In both cases, higher curves correspond to better performance.

non-linear function approximation. The gap between QU-SAC and the distributional methods (QR-MBPO / EQR-SAC) indicates that the quantile representation of values leads to more sample-efficient learning compared to the ensemble-based approach. Moreover, training one distributional critic is typically less computationally intensive than training an ensemble of standard critics. In the next section, we investigate more deeply the performance difference between EQR-SAC and QR-MBPO.

3.7.4. Why Does EQR-SAC Outperform QR-MBPO?

We conduct an additional experiment to determine what component(s) of EQR-SAC are responsible for the increased performance with respect to QR-MBPO. There is three differences between EQR-SAC and QR-MBPO: (i) EQR-SAC has a buffer n times bigger than QR-MBPO, and correspondingly scales up the amount of data collected under the ensemble, (ii) EQR-SAC uses consistent rollouts, while QR-MBPO randomizes the model per-step, and (iii), EQR-SAC's critic is trained on the loss (3.19), while QR-MBPO's critic uses (3.22). In order to test the impact of each component in isolation, we add three additional baselines: QR-MBPO-big, which uses the same buffer size and collects the same amount of data as EQR-SAC; QR-MBPO-consistent, that replicates how EQR-SAC collects data under the model; and QR-MBPO-loss, that uses (3.19) to train its critic. Figure 3.11 shows the performance of EQR-SAC and all QR-MBPO variants (all methods optimize the actor using f_{mean}). The main observation is that QR-MBPO-loss matches closely the performance of EQR-SAC, while all other QR-MBPO variants share similar (lower) performance. The key insight from these results is that our proposed critic loss function (3.19) is instrumental towards sample-efficient learning, especially in environments with sparse rewards like cartpole swing-up (see also fish-swim and finger-spin in Appendix B.3). As such, our theory provides a solid guideline on how to integrate model-based RL architectures with distributional RL tools, which goes beyond simply using a distributional critic with established algorithms like MBPO.

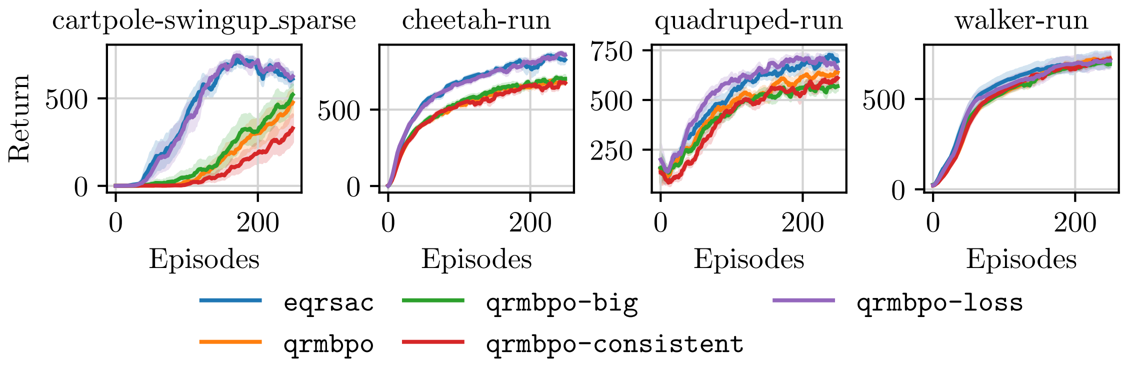


Figure 3.11.: Comparison of EQR-SAC and QR-MBPO baselines in selected DeepMind Control tasks. The results suggest that the biggest contributing factor for increased performance of EQR-SAC w.r.t QR-MBPO is the critic's loss function (3.19).

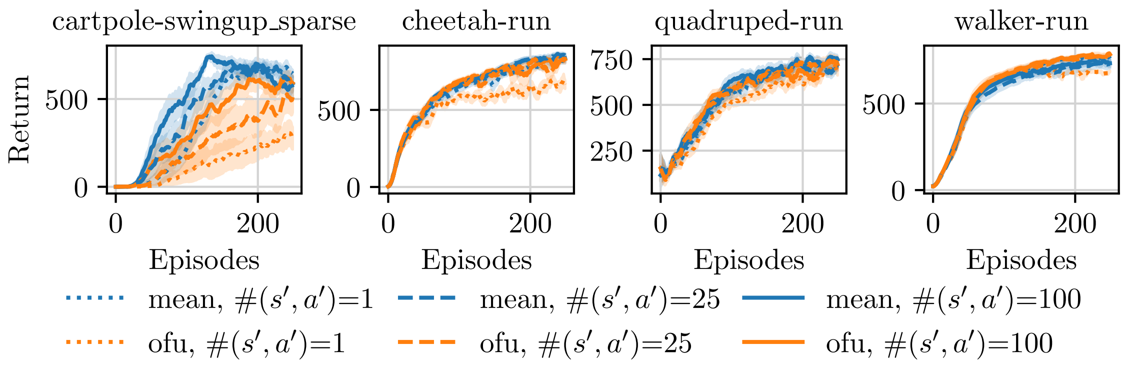


Figure 3.12.: Ablation study on the amount of next state-action samples drawn to approximate the target quantiles (3.20). Larger sample sizes perform more robustly across all environments.

3.7.5. Dynamics Sampling for Target Quantiles

The analysis in Section 3.7.4 points to the loss function (3.19) as being the key component of our proposed approach. The main feature of our loss function is how it utilizes the generative model to produce the target quantiles (3.20). In this experiment, we investigate the effect of the amount of next state-action samples (s', a') drawn from the ensemble of dynamics when estimating the target quantiles. The hypothesis is that a larger sample size would result in a lower-variance estimate of the expectation in (3.20), which could then lead to better sample-efficiency. Figure 3.12 shows the performance of EQR-SAC, for both f_{mean} and f_{ofu}, under different sampling regimes. For the cartpole task, we observe a clear progression in sample-efficiency as the amount of samples increases. For other environments the differences are less noticeable, but using a sample size of 1 with the optimistic objective leads to worse performance in all cases. Since the sample size might have large effects in performance and its runtime impact is greatly amortized by GPU parallelization, the overall recommendation is to use larger sample sizes (25-100) by default.

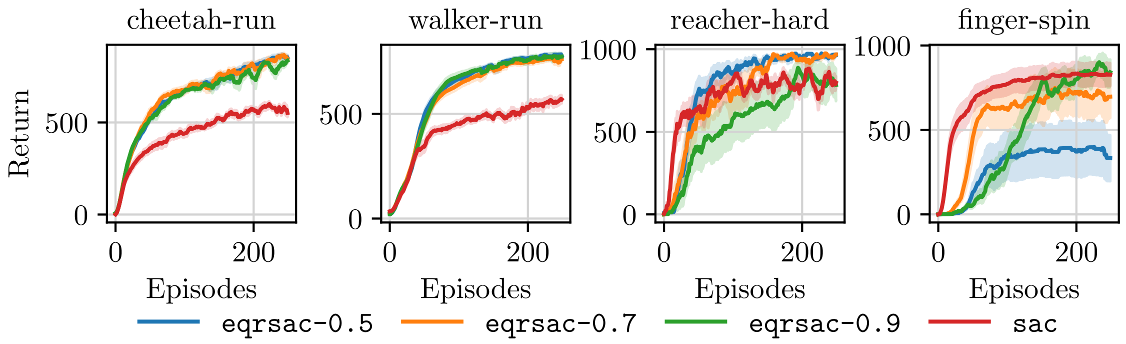


Figure 3.13.: Evaluation of EQR-SAC-τ for different quantile levels. The two tasks on the left have dense rewards, while the other two have sparse rewards.

3.7.6. Optimistic Policy Optimization

We investigate the effect of using optimistic value estimates for policy optimization. To conduct the study, we propose a simple variant of EQR-SAC, named EQR-SAC-τ, which uses as the actor's objective function the closest quantile level to a given target τ. For instance, in our experiments we use $m = 21$ and target levels $\{0.5, 0.7, 0.9\}$, which correspond to actual levels $\{0.5, 0.69, 0.88\}$.

Dense versus sparse rewards. We first investigate how optimism affects performance in environments with dense versus sparse rewards and present the results in Figure 3.13. For environments with dense rewards (cheetah, walker) optimism has little to no effect, while it results in largely different performance in envionments with sparse rewards (reacher-hard, finger-spin). Even though we would expect optimism to be generally helpful in all exploration tasks, our results indicate its effect is environment-dependent: the most optimistic objective ($\tau = 0.9$) performed worst in reacher-hard but obtained the best performance in finger-spin; inversely, the least optimistic objective ($\tau = 0.5$) performed the best in reacher-hard, but worst in finger-spin.

Action costs and sparse rewards. In Section 3.7.2, we observe that the combination of action costs and sparse rewards represents a pitfall for methods like SAC, especially since the optimal policy must issue large actions to observe the reward. Meanwhile, the quantile-based optimistic approaches performed best. In this experiment, we test the same setting in two tasks with sparse rewards from the DeepMind Control suite, where we add an action cost proportional to the squared norm of the action taken by the agent. Namely,

$$\texttt{action\_cost} = \rho \sum_{i=1}^{|\mathcal{A}|} a_i^2 \tag{3.24}$$

where ρ is an environment specific base multiplier, a_i is the i-th component of the action vector and $|\mathcal{A}|$ is the size of the action space. For `cartpole-swingup` we use $\rho = 0.001$ and for `pendulum` we use $\rho = 0.01$. The resuls in Figure 3.14 show a similar degradation of performance for SAC. Unlike the MountainCar experiments of Section 3.7.2, higher levels of optimism mostly resulted in less sample-efficient learning.

Overall, all these results indicate that the benefits of optimistic optimization might be environment-dependent. We believe an interesting avenue for future work is to more broadly analyze this phenomenon and reconsider our design choices (ensemble as posterior MDP, quantile representation for values, policy objectives, etc.).

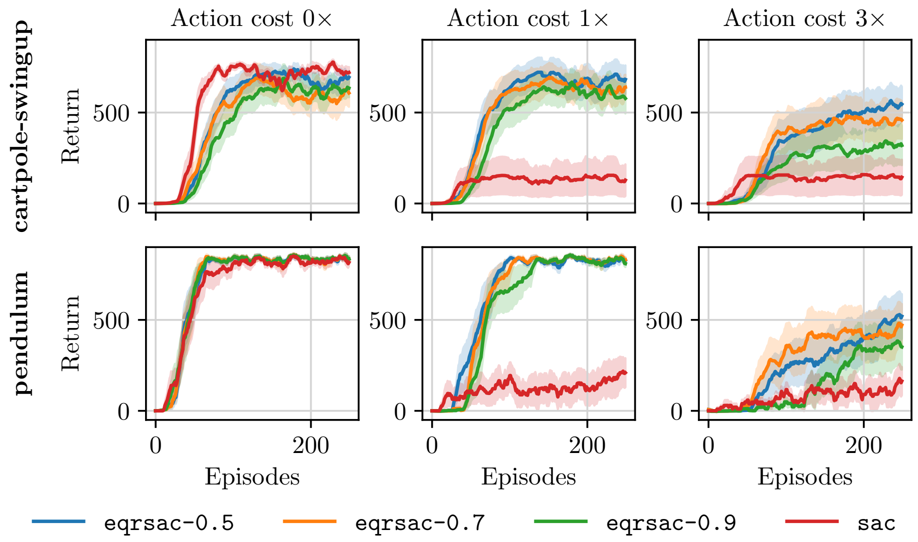


Figure 3.14.: Evaluation of EQR-SAC-τ for different quantile levels and increasing action costs. The top row corresponds to the cartpole swingup task and the bottom row to the pendulum swingup. The action costs range from zero (left column), to a $3\times$ multiplier on (3.24).

3.7.7. Additional Ablations

We conduct additional ablation studies on three hyperparameters of our algorithm: the number of quantiles (m) (Figure 3.15), the rollout length (k) (Figure 3.16) and the number of model updates to retain data (Δ), which controls the amount of off-policy data in the buffer (Figure 3.17). The general observation from these experiments is that EQR-SAC's performance is robust for a wide range of values. Performance typically degrades only for extreme values of the parameters, for example $m = 1$ (only estimate median of value distribution) or $k = 1$ (only generate 1-step rollouts with the model).

3.8. Conclusions

We investigated the problem of estimating the distribution of values, given parameter uncertainty of the MDP. We proposed the value-distributional Bellman equation and extracted an operator whose fixed-point is precisely the distribution of values. Leveraging tools from return-distributional RL, we designed Epistemic Quantile-Regression, an iterative procedure for estimating quantiles of the value distribution. We applied our algorithm in small MDPs, validated the convergence properties prescribed by our theory and assessed its limitations once the main assumptions are violated. Lastly, we introduced EQR-SAC, a novel model-based deep RL algorithm that scales up EQR with neural network function approximation and combines it with SAC for policy optimization. We benchmarked our approach in several continuous-control tasks from the DeepMind Control suite and showed improved sample-efficiency and final performance compared to various baselines.

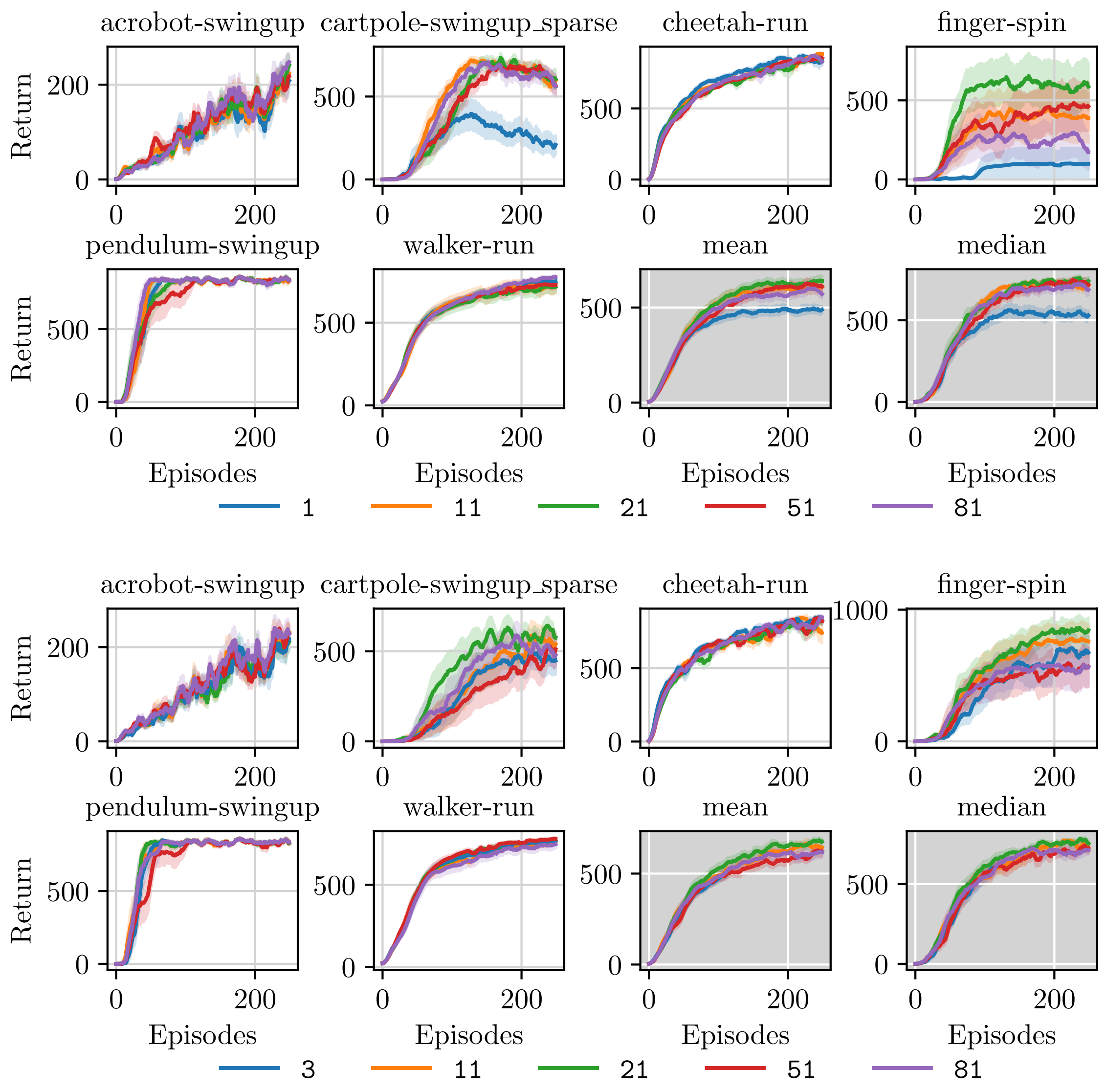


Figure 3.15.: Number of quantiles (m) ablation study. **(Top)** EQR-SAC-mean. **(Bottom)** EQR-SAC-ofu. Note that for EQR-SAC-ofu we require $m > 1$ in order to estimate the standard deviation of quantiles for the optimistic objective function of the actor, thus we select a minimum value of $m = 3$ for this study.

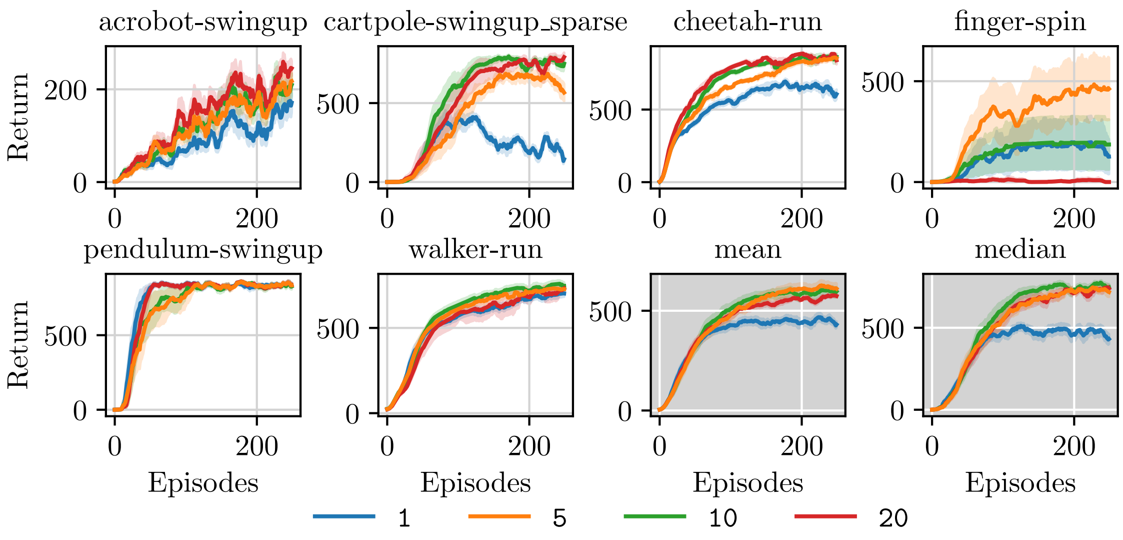


Figure 3.16.: Rollout length (k) ablation study for EQR-SAC-mean.

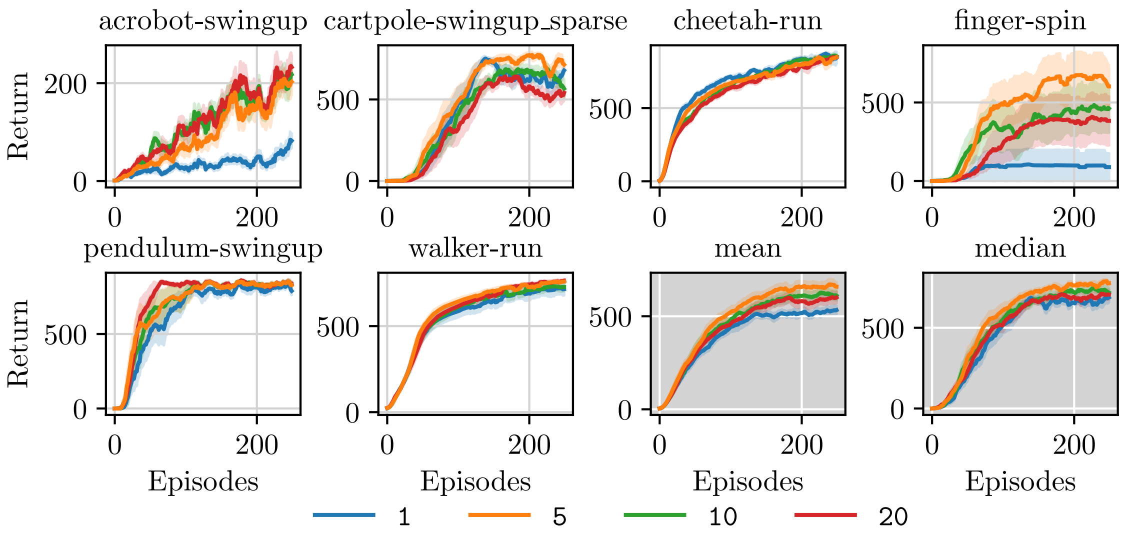


Figure 3.17.: Number of model updates to retain data (Δ) ablation study for EQR-SAC-mean.

4. Uncertainty Representations in State-Space Layers for Deep Reinforcement Learning under Partial Observability

Optimal decision-making under partial observability requires reasoning about the uncertainty of the environment's hidden state. However, most reinforcement learning architectures handle partial observability with sequence models that have no internal mechanism to incorporate uncertainty in their hidden state representation, such as recurrent neural networks, deterministic state-space models and transformers. Inspired by advances in probabilistic world models for reinforcement learning, we propose a standalone Kalman filter layer that performs closed-form Gaussian inference in linear state-space models and train it end-to-end within a model-free architecture to maximize returns. Similar to efficient linear recurrent layers, the Kalman filter layer processes sequential data using a parallel scan, which scales logarithmically with the sequence length. By design, Kalman filter layers are a drop-in replacement for other recurrent layers in standard model-free architectures, but importantly they include an explicit mechanism for probabilistic filtering of the latent state representation. Experiments in a wide variety of tasks with partial observability show that Kalman filter layers excel in problems where uncertainty reasoning is key for decision-making, outperforming recurrent other stateful models.

4.1. Introduction

The classical reinforcement learning (RL) formulation tackles optimal decision-making in a fully observable Markov Decision Process (MDP) (Sutton and Barto, 2018). However, many real-world problems are *partially* observable, since we only have access to observations that hide information about the state, e.g., due to noisy measurements. Learning in partially observable MDPs (POMDPs) is statistically and computationally intractable in general (Papadimitriou and Tsitsiklis, 1987), but in many practical scenarios it is theoretically viable (Liu et al., 2022) and has lead to successful applications in complex domains like robotics (Zhu et al., 2017), poker (Brown and Sandholm, 2019), real-time strategy games (Vinyals et al., 2019) and recommendation systems (Li et al., 2010).

Practical algorithms for RL in POMDPs employ sequence models to encode the history of observations and actions into a latent state representation amenable for policy optimization. Besides extracting task-relevant information from the history, probabilistic inference over the latent state is also crucial under partial observability (Kaelbling et al., 1998). As a motivating example, consider an AI chatbot that gives restaurant recommendations to users. Since the user's taste (i.e., the state) is unknown, the agent must ask questions before ultimately making its recommendation. Reasoning over the latent state uncertainty is crucial to decide whether to continue probing the user or end the interaction with a final recommendation. An optimal agent would gather enough information to recommend a restaurant with a high likelihood of user satisfaction. In Section 4.5.3, we evaluate performance of our proposed approach in a simplified version of this problem.

A standard recipe for model-free RL in POMDPs is to combine a sequence model (e.g., LSTM (Hochreiter and Schmidhuber, 1997), GRU (Cho et al., 2014)) with a policy optimizer (e.g., PPO (Schulman et al., 2017) or SAC (Haarnoja et al., 2018)), which has shown strong performance in a wide variety of POMDPs (Ni et al., 2022). More recently, transformers (Vaswani et al., 2017) have also been adopted as sequence models in RL showing improved memory capabilities (Ni et al., 2023). However, their inference runtime scales quadratically with the sequence length, which makes them unsuitable for online learning in physical systems (Parisotto and Salakhutdinov, 2020). Instead, recent deterministic state-space models (SSMs) (Gu et al., 2022a; Smith et al., 2023; Gu and Dao, 2023) maintain the constant-time inference of stateful models, while achieving logarithmic runtime during training thanks to efficient parallel scans (Smith et al., 2023). Moreover, SSMs have shown improved long-term memory, in-context learning and generalization in RL (Lu et al., 2023). Yet, in problems where reasoning over latent state uncertainty is crucial, it remains unclear whether such methods can learn the required probabilistic inference mechanisms for decision making.

While model-free architectures focus on *deterministic* sequence models, in model-based RL *probabilistic* sequence models are a widespread tool to model uncertainty in environment dynamics (Hafner et al., 2019, 2020; Becker and Neumann, 2022). Considering these two sequence modelling approaches, the purpose of this chapter is to investigate the following questions:

Can we leverage the same inference methods developed for model-based RL as general-purpose sequence models in model-free architectures? If so, does it bring any benefits compared to deterministic models?

Our core hypothesis is that explicit probabilistic inference in sequence models may serve as an inductive bias to learn in tasks where uncertainty over the latent state is crucial for decision making, as our previous example on the restaurant recommendation chatbot.

Our Contributions. Inspired by the simple inference scheme in the Recurrent Kalman Network (RKN) (Becker et al., 2019) architecture for world models, we embed closed-form Gaussian inference in linear SSMs as a standalone recurrent layer — denoted a Kalman filter (KF) layer — and train it end-to-end within a model-free architecture (Ni et al., 2022) to maximize returns. Since our KF layers are designed to be a drop-in replacement for standard recurrent layers, they can also be stacked together and combined with other components (e.g., residual connections, normalization, etc.) to build more complex sequence models. To the best of our knowledge, this is the first work that empirically evaluates the performance of such probabilistic inference layers in a model-free RL architecture. We conduct extensive experiments with a variety of sequence models over a wide range of POMDPs and show that KF layers excel in tasks where probabilistic inference is key for decision-making, with significant improvements over deterministic stateful models.

4.2. Related Work

RL architectures for POMDPs. In order to deal with partial observability, RL agents are typically equipped with some form of *memory* system, e.g., based on human psycology (Fortunato et al., 2019) or context-dependent retrieval (Oh et al., 2016). The most widespread memory system in the RL literature is through *sequence models*, also known as *history encoders* (Ni et al., 2024), whose purpose is to *compress* the past history into a state representation useful for RL. These sequence models can augment policies (Wierstra et al., 2007), value functions (Schmidhuber, 1990; Bakker, 2001) and/or world models (Schmidhuber, 1991; Becker et al., 2019; Shaj et al., 2021a,b, 2023), which allows handling of partial observability in previous RL algorithms such as DQN (Hausknecht and Stone, 2015), SAC (Ni et al., 2022), PPO (Kostrikov, 2018; Ni et al., 2023; Lu et al., 2023), DPG (Heess et al., 2015) and Dyna (Hafner et al., 2020; Becker and Neumann, 2022). In this chapter, we adopt an off-policy model-free architecture similar to Ni et al. (2022), which showed strong performance in various POMDPs.

Sequence models in RL. Frame-stacking is one of the earliest sequence models used in RL (Lin and Mitchell, 1993) and it is still a common tool to convey velocity information from image-based observations, like in the Atari benchmark (Bellemare et al., 2013; Mnih et al., 2013). However, it fails to model long-range dependencies in more complex POMDPs. Stateful recurrent models are the most widespread sequence models in RL to extract relevant information from arbitrarily long contexts, for instance RNNs (Lin and Mitchell, 1993; Schmidhuber, 1990) LSTMs (Bakker, 2001) and GRUs (Kostrikov, 2018). More recently, the transformer architecture (Vaswani et al., 2017) has shown promising results in improving the long-term memory of RL agents (Ni et al., 2023). However, their slow inference and large memory requirements reduce their practicality as real-time control systems (Parisotto and Salakhutdinov, 2020).

Deterministic SSMs. Advances in SSMs are of particular interest to the RL community, since they maintain the fast inference of RNNs but scale better during training via parallel scans (Smith et al., 2023), can circumvent vanishing/exploding gradients with proper initialization (Gu et al., 2020) and match (or even exceed) the performance of transformers in long-range sequence modelling tasks (Lu et al., 2023). In particular, *structured* state space models such as S4 (Gu et al., 2022a), S5 (Smith et al., 2023) and S6 / Mamba (Gu and Dao, 2023) have emerged as a strong competitor to transformers in general sequence modelling problems like language (Fu et al., 2023), audio (Goel et al., 2022) and video (Nguyen et al., 2022). The adoption of these models in RL is still in its infancy, however. Morad et al. (2023) report bad performance of a variant of S4 (Gu et al., 2022b) in various POMDPs. Lu et al. (2023) show strong performance in long-term memory and in-context learning by combining S5 with PPO. Besides these contradictory results, it remains unclear how these models perform in tasks where uncertainty of the latent state plays a vital role in decision-making since they are not equipped with explicit probabilistic inference mechanisms.

Probabilistic SSMs. Probabilistic stateful models are the backbone of world models (Kalweit and Boedecker, 2017; Ha and Schmidhuber, 2018), trained to predict forward dynamics and rewards which can then be used for: (i) planning (Hafner et al., 2019) or policy optimization (Becker and Neumann, 2022; Hafner et al., 2020) via latent imagination (i.e., generate imaginary policy rollouts auto-regressively), or (ii) policy optimization on the learned latent representation (Becker et al., 2024). A prominent approach is the Recurrent State Space Model (RSSM) proposed by Hafner et al. (2019), which divides the latent state into deterministic and stochastic components and uses a GRU for propagating forward the deterministic part. More recently, GRUs have been replaced by transformers (Chen et al., 2021) and S4 (Samsami et al., 2024) models, albeit in a simplified inference scheme that only conditions on the current observation rather than the history. A common concern with these model-based approaches is the objective mismatch (Lambert et al., 2020) between learning accurate world models and training performant control policies.

Inference in Linear SSMs. Closest to our approach are models that perform closed-form probabilistic filtering in *linear* SSMs. Inference in linear SSMs is a well-studied problem in the dynamical systems community, dating back to the seminal work by Kalman (1960). While Kalman filters and smoothers offer tractable, closed-form inference, the linear-Gaussian assumption is typically not suited for high-dimensional, multi-modal data (Murphy, 2012; Bishop, 2006). Recent methods leverage neural networks to project data into learned latent spaces where the Kalman filter assumptions are less restrictive, leading to expressive probabilistic models of high-dimensional data such as images (Haarnoja et al., 2016; Becker et al., 2019;

Shaj et al., 2021a,b, 2023). The Recurrent Kalman Network (RKN) proposed by Becker et al. (2019) is an encoder-decoder architecture that employs Kalman filtering using locally linear models and structured (non-diagonal) covariance matrices. Follow-up work has extended the RKN framework in various ways: Shaj et al. (2021a) include action conditioning, Shaj et al. (2021b) consider a multi-task setting with hidden task parameters, Shaj et al. (2023) propose a hierarchical, multi-timescale architecture and Becker and Neumann (2022) replace the closed-form filtering with a variational approach that also performs smoothing, leading to a tight variational bound. Concurrent work by Becker et al. (2024) proposes a world model that leverages a Mamba (Gu and Dao, 2023) backbone to learn a linear dynamics model and a time-parallel Kalman smoother trained on a variational inference loss; the learned state representation is then used with a SAC policy optimizer.

In this chapter, we propose a model-free RL architecture for POMDPs, similar to Ni et al. (2022), where deterministic recurrent layers (e.g., RNNs or transformers) are replaced with KF layers that leverage the simplified filtering scheme from RKNs. In contrast to prior work proposing RKN-based architectures for model-based RL (Becker and Neumann, 2022; Becker et al., 2024), in our model-free RL architecture KF layers are trained end-to-end to maximize returns instead of auxiliary world model objectives. Consequently, our training paradigm erases concerns of objective mismatch during representation learning (Lambert et al., 2020). Moreover, we design KF layers as standalone recurrent layers that can be stacked and combined with other operations (e.g., residual connections and normalization) to build more complex architectures. Similar to Becker et al. (2024), the associative property of the Kalman filter operations allows efficient training of KF layers via parallel scans, which scale logarithmically with the sequence length provided sufficient parallel GPU cores.

4.3. Background

In this section, we provide the relevant background and introduce core notation used throughout the chapter. We use bold upper case letters ($\mathbf{A}$) to denote matrices and calligraphic letters ($\mathcal{X}$) to denote sets. The notation $\mathcal{P}(\mathcal{X})$ referes to the space of probability distributions over $\mathcal{X}$.

4.3.1. Reinforcement Learning in Partially Observable Markov Decision Processes

We consider an agent that acts in a finite-horizon partially observable Markov decision process (POMDP) $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{O}, T, p, O, r, \gamma\}$ with state space $\mathcal{S}$, action space $\mathcal{A}$, observation space $\mathcal{O}$, horizon $T \in \mathbb{N}$, transition function $p : \mathcal{S} \times \mathcal{A} \to \mathcal{P}(\mathcal{S})$ that maps states and actions to a probability distribution over $\mathcal{S}$, an emission function $O : \mathcal{S} \to \mathcal{P}(\mathcal{O})$ that maps states to a probability distribution over observations, a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and a discount factor $\gamma \in [0, 1)$.

At time step t of an episode in $\mathcal{M}$, the agent observes $o_t \sim O(\cdot \mid s_t)$ and selects an action $a_t \in \mathcal{A}$ based on the observed history $h_{:t} = \left\{(o_h, a_h)\right\}_{h=0}^{t} \in \mathcal{H}_t$, then receives a reward $r_t = r(s_t, a_t)$ and the next observation $o_{:t+1} \sim O(\cdot \mid s_{t+1})$ with $s_{t+1} \sim p(\cdot \mid s_t, a_t)$.

We adopt the general setting by Ni et al. (2023, 2024), where the RL agent is equipped with: (*i*) a stochastic policy $\pi : \mathcal{H}_t \to \mathcal{P}(\mathcal{A})$ that maps from observed history to distribution over actions, and (*ii*) a value function $Q^\pi : \mathcal{H}_t \times \mathcal{A} \to \mathbb{R}$ that maps from history and action to the expected return under the policy, defined as $Q^\pi(h_{:t}, a_t) = \mathbb{E}_\pi\left[\sum_{h=t}^{T} \gamma^{h-t} r_t \mid h_{:t}, a_t\right]$. The objective of the agent is to find the optimal policy that maximizes the value starting from some initial state s_0, $\pi^\star = \operatorname{argmax}_\pi \mathbb{E}_\pi\left[\sum_{t=0}^{T-1} \gamma^t r_t \mid s_0\right]$.

4.3.2. History Representations

A weakness of the general formulation of RL in POMDPs is the dependence of both the policy and the value function on the ever-growing history. Instead, practical algorithms fight this curse of dimensionality by encoding the history into a compact representation. Ni et al. (2024) propose to learn such representations via *history encoders*, defined by a mapping $\phi : \mathcal{H}_t \to \mathcal{Z}$ from observed history to some latent representation $z_t := \phi(h_{:t}) \in \mathcal{Z}$. With slight abuse of notation, we denote $\pi(a_t \mid z_t)$ and $Q^\pi(z_t, a_t)$ as the policy and values under this latent representation, respectively. In this paper, we propose a history encoder implemented via Kalman filtering layers that perform simple probabilistic inference on the latent state.

4.3.3. Structured State Space Models

We consider time-varying linear SSMs defined by

$$\dot{x}(t) = \mathbf{A}_t x(t) + \mathbf{B}_t u(t), \quad y(t) = \mathbf{C}_t z(t) + \mathbf{D}_t u(t), \tag{4.1}$$

where $t > 0 \in \mathbb{R}$, $x(t) \in \mathbb{R}^N$ is the hidden or latent state, $u(t) \in \mathbb{R}^P$ is the input, $y(t) \in \mathbb{R}^M$ is the output and $(\mathbf{A}_t, \mathbf{B}_t, \mathbf{C}_t, \mathbf{D}_t)$ are matrices of appropriate size. Such a continuous-time system can be discretized (e.g., using zero-order hold) for some step size Δ, which results in a linear recurrent model

$$x_k = \bar{\mathbf{A}}_k x_{k-1} + \bar{\mathbf{B}}_k u_{k-1}, \quad y_k = \bar{\mathbf{C}}_k x_k + \bar{\mathbf{D}}_k u_{k-1}. \tag{4.2}$$

As it is common in practice, we set $\bar{\mathbf{D}}_k \equiv \mathbf{0}$. In this chapter, we consider *structured* SSMs, which simply means special structure is imposed into the learnable matrices $(\bar{\mathbf{A}}_k, \bar{\mathbf{B}}_k, \bar{\mathbf{C}}_k)$. In particular, we consider a diagonal structure with the HiPPO initialization proposed in Gu et al. (2020), which induces stability in the recurrence for handling long sequences.

4.3.4. Probabilistic Inference on Linear SSMs

To introduce uncertainty into state-space models, we consider a standard linear-Gaussian SSM

$$x_k = \bar{\mathbf{A}}_k x_{k-1} + \bar{\mathbf{B}}_k u_{k-1} + \varepsilon_k, \quad y_k = \bar{\mathbf{C}}_k x_k + \nu_k, \tag{4.3}$$

where $\varepsilon_k \sim \mathcal{N}(0, \mathbf{\Sigma}_k^{\mathrm{p}})$ and $\nu_k \sim \mathcal{N}(0, \mathbf{\Sigma}_k^{\mathrm{o}})$ are zero-mean process and observation noise variables with their covariance matrices $\mathbf{\Sigma}_k^{\mathrm{p}}$ and $\mathbf{\Sigma}_k^{\mathrm{o}}$, respectively. The latent state probabilistic model is then $p(x_k \mid x_{k-1}, u_{k-1}) = \mathcal{N}(\bar{\mathbf{A}}_k x_{k-1} + \bar{\mathbf{B}}_k u_{k-1}, \mathbf{\Sigma}_k^{\mathrm{p}})$ and the observation model is $p(y_k \mid x_k) = \mathcal{N}(\bar{\mathbf{C}}_k x_k, \mathbf{\Sigma}_k^{\mathrm{o}})$. Inference in such a model has a closed-form solution, which is equivalent to the well-studied Kalman filter (Kalman, 1960).

Predict. The first stage of the Kalman filter propagates forward the *posterior* belief of the latent state at step $k-1$, given by $\mathcal{N}(x_{k-1}^+, \mathbf{\Sigma}_{k-1}^+)$, to obtain a *prior* belief at step k, $\mathcal{N}(x_k^-, \mathbf{\Sigma}_k^-)$, given by

$$x_k^- = \bar{\mathbf{A}}_k x_{k-1}^+ + \bar{\mathbf{B}}_k u_{k-1}, \quad \mathbf{\Sigma}_k^- = \bar{\mathbf{A}}_k \mathbf{\Sigma}_{k-1}^+ \bar{\mathbf{A}}_k^\top + \mathbf{\Sigma}_k^{\mathrm{p}}. \tag{4.4}$$

Update. The second stage updates the prior belief at step k given some observation w_k, to obtain the posterior $p(x_k \mid x_{k-1}, w_k) = \mathcal{N}(x_k^+, \mathbf{\Sigma}_k^+)$ given by

$$x_k^+ = x_k^- + \mathbf{K}_k(w_k - \bar{\mathbf{C}}_k x_k^-), \quad \mathbf{\Sigma}_k^+ = (\mathbf{I} - \mathbf{K}_k \bar{\mathbf{C}}_k)\mathbf{\Sigma}_k^-, \tag{4.5}$$

where $\mathbf{K}_k = \mathbf{\Sigma}_k^- \bar{\mathbf{C}}_k^\top (\bar{\mathbf{C}}_k \mathbf{\Sigma}_k^- \bar{\mathbf{C}}_k^\top + \mathbf{\Sigma}_k^{\mathrm{o}})^{-1}$ is known as the Kalman gain. The predict and update steps are interleaved to process sequences of input and observations $\{u_k, w_k\}_{k=0}^{K-1}$ of length K, starting from some initial belief $\mathcal{N}(x_{-1}^+, \mathbf{\Sigma}_{-1}^+)$. While (4.4) and (4.5) include expensive matrix operations, they simplify to cheap element-wise operations under structured SSMs (e.g., diagonal), as done in prior work (Becker et al., 2019; Becker and Neumann, 2022).

4.3.5. Parallel Scans

Efficient implementation of state-space models and Kalman filters employ *parallel scans* to achieve logarithmic runtime scaling with the sequence length (Smith et al., 2023; Sarkka and Garcia-Fernandez, 2021). Given a sequence of elements $(a_0, a_1, \ldots, a_{K-1})$ and an *associative*[1] binary operator $\bullet$, the parallel scan algorithm outputs all the prefix-sums $(a_0, a_0 \bullet a_1, \ldots, a_0 \bullet \ldots \bullet a_{K-1})$ in $\mathcal{O}(\log K)$ runtime, given sufficient parallel processors.

4.4. Method: Off-Policy Recurrent Actor-Critic with Kalman filter Layers

In this section, we describe our method that implements Kalman filtering as a recurrent layer within a standard actor-critic architecture.

4.4.1. General Architecture

In Figure 4.1 we present our Recurrent Actor-Critic (RAC) architecture inspired by Ni et al. (2022), where we replace the RNN blocks with general history encoders. We will use this architecture in the following to test the capabilities of different history encoders in various POMDPs.

For both actor and critic, we embed the sequence of observations and actions into a single representation $h_{:t}^*$ which is then passed into the history encoders. We use a single linear layer as embedder, which we found worked as reliably as more complex non-linear embedders used in similar RAC architectures by Morad et al. (2023); Ni et al. (2022). We also include the skip connections from current observations and actions into the actor-critic heads, as proposed in previous memory-based architectures (Zintgraf et al., 2021; Ni et al., 2022).

[1]A binary operator $\bullet$ is associative if $(a \bullet b) \bullet c = a \bullet (b \bullet c)$ for any triplet of elements (a, b, c)

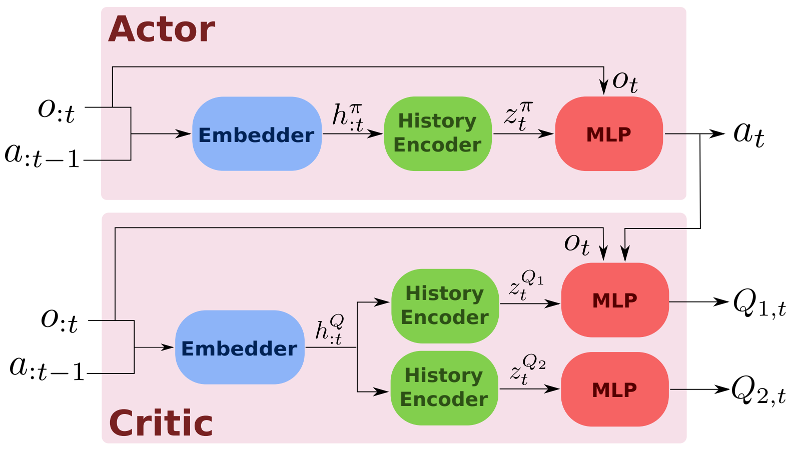


Figure 4.1.: General Recurrent Actor-Critic (RAC) architecture. The components are trained end-to-end with the Soft Actor-Critic (SAC) loss function (Haarnoja et al., 2018). To handle discrete action spaces, we use the discrete version of SAC by Christodoulou (2019).

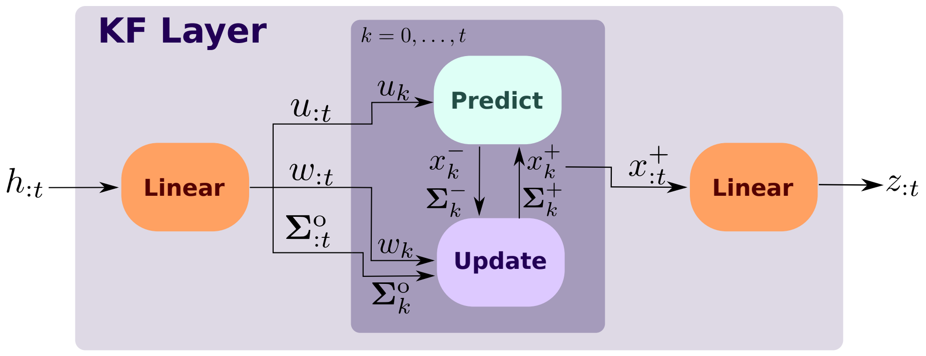


Figure 4.2.: Our proposed Kalman filter layer to build history encoders. The KF layer receives a history sequence $h_{:t}$ and projects it into three separate signals in latent space: the input $u_{:t}$, the observation $w_{:t}$ and the observation noise (diagonal) covariance $\Sigma^{\mathrm{o}}_{:t}$. These sequences are processed using the standard Kalman filtering equations, which scale logarithmically with the sequence length using parallel scans. Lastly, the posterior mean latent state $x^+_{:t}$ is projected from the latent space back into the history space to obtain the compressed representation $z_{:t}$.

4.4.2. Kalman Filter Layers

Our main hypothesis is that principled probabilistic filtering within history encoders boosts performance in POMDPs, especially those where reasoning about uncertainty is key for decision-making. To test this hypothesis, we introduce KF layers, as shown in Figure 4.2. The layer receives as input a history embedding sequence $h_{:t}$ which is then projected into the input $u_{:t}$, observation $w_{:t}$ and observation noise $\Sigma^{\mathrm{o}}_{:t}$ sequences. These three signals serve as input to the standard KF predict-update equations (4.4) and (4.5), which output a posterior (filtered) latent state $x^+_{:t}$. Finally, the posterior sequence is projected back to the history embedding space to produce the compressed history representation $z_{:t}$.

History encoders with KF layers. Similar to recent SSM layers such as S5 (Smith et al., 2023) and S6 (Gu and Dao, 2023), these KF layers can be stacked and combined with other operations such as residual connections, gating mechanisms, convolutions and normalization to compose a history encoder block in the

RAC architecture. In favor of simplicity, our history encoders are only composed of KF layers and (optionally) an RMS normalization (Zhang and Sennrich, 2019) output block for improved stability.

Filtering as a gating mechanism. We can draw interesting comparisons between KF layers and other recurrent layers from the perspective of gating mechanisms. It was shown in Theorem 1 of (Gu and Dao, 2023) that selective SSMs (S6) behave as generalized RNN gates through an input-dependent step size Δ. In this case, the gate depends on the SSM input and controls how much the input influences the next hidden state. Similarly, as hinted by Becker et al. (2019), during the update step the Kalman gain is effectively an uncertainty-controlled gate depending on the observation noise which regulates how much the observation influences the posterior belief over the latent state. Our experiments in Section 4.5 shed some light on the strengths and weaknesses of these approaches for RL under partial observability.

Implementation details. We simplify the implementation of KF layers and eliminate expensive matrix operations via a diagonal-structured SSM with matching input-latent-observation dimensions, i.e., $N = P = M$. In this setting, we directly learn the diagonal components of time-independent matrices $\mathbf{A}$ and $\mathbf{B}$ (no diagonalization step as done in the S5 layer (Smith et al., 2023)), while we fix $\mathbf{C}$ to be the identity matrix, as it is common in prior work (Becker and Neumann, 2022). Second, we consider a time-invariant process noise with learnable diagonal covariance matrix $\mathbf{\Sigma}^{\mathrm{p}}$. The SSM is discretized using the zero-order hold method and consider a learnable scalar step size $\Delta > 0$ (Smith et al., 2023).

Design decisions. We want to highlight two considerations that went into the design of our KF layers. First, we could generalize the architecture to support time-varying process noise by including one extra output channel (alongside the input, observation and observation noise channels) in the history linear projection. Conceptually, such an input-dependent process noise adds more flexibility to the gating mechanism implemented within the KF layer, which would be controlled both by the observation and the process noise signals. Second, we could include the posterior covariance $\mathbf{\Sigma}^{+}_{:t}$ as an additional feature for the output linear projection, alongside the posterior mean $x^{+}_{:t}$. We conduct an ablation study over these two choices in several continuous control tasks subject to observation noise and report the results in Appendix C.3. The best aggregated performance in this ablation was obtained with time-invariant process noise and only using the posterior mean as a feature for the output projection, which empirically justifies our final design.

4.4.3. Masked Associative Operators for Variable Sequence Lengths

In off-policy RAC architectures, the agent is typically trained with batches of (sub-)trajectories of possibly different length, sampled from an experience replay buffer. Thus, history encoders must be able to process batches of variable sequence length during training.

A common approach is to right-pad the batch of sequences up to a common length and ensure the model's output is independent of the padding values. For transformer models, this can be achieved by using the padding mask as a self-attention mask. For stateful models like RNNs and SSMs, it is imperative to also output the correct final latent state for each sequence in the batch. This typically requires a post-processing step that individually selects for each sequence in the batch the last state before padding. It turns out that for any recurrent model expressed with an associative operator (e.g., SSMs and KFs), we can obtain the correct final state from a batch of padded sequences *without* additional post-processing by using a parallel scan routine with a *Masked Associative Operator* (MAO).

Definition 4 (Masked Associative Operator)**.** Let $\bullet$ be an associative operator acting on elements $e \in \mathcal{E}$, such that for any $a, b, c \in \mathcal{E}$, it holds that $(a \bullet b) \bullet c = a \bullet (b \bullet c)$. Then, the MAO associated with $\bullet$, denoted

$\tilde{\bullet}$, acts on elements $\tilde{e} \in \mathcal{E} \times \{0,1\} = (e, m)$, where $m \in \{0,1\}$ is a binary mask. Then, for $\tilde{a} = (a, m_a)$ and $\tilde{b} = (b, m_b)$, we have:

$$\tilde{a} \;\tilde{\bullet}\; \tilde{b} = \begin{cases} (a \bullet b, m_a) & \text{if} \quad m_b = 0 \\ \tilde{a} & \text{if} \quad m_b = 1 \end{cases} \tag{4.6}$$

Now we show that any MAO is itself *associative* as long as we apply a right-padding mask[2].

Proof. Let $\tilde{a}, \tilde{b}, \tilde{c} \in \tilde{\mathcal{E}}$ refer to elements in the space of the MAO $\tilde{\bullet}$, as in Definition 4, with $\tilde{a} = (a, m_a)$, $\tilde{b} = (b, m_b)$, $\tilde{c} = (c, m_c)$. We show that if the sequence $\{m_a, m_b, m_c\}$, is a right-padding mask, that is: $m_a = 1 \implies m_b = m_c = 1$, and $m_b = 1 \implies m_c = 1$, then it holds that $(\tilde{a} \;\tilde{\bullet}\; \tilde{b}) \;\tilde{\bullet}\; \tilde{c} = \tilde{a} \;\tilde{\bullet}\; (\tilde{b} \;\tilde{\bullet}\; \tilde{c})$, i.e., the MAO is associative. Similar to the proof in Lu et al. (2023) we consider all possible values for $\{m_a, m_b, m_c\}$.

Case 1: $m_b = 1$ and $m_c = 1$. The binary masks of b and c are on, so $\tilde{b} \;\tilde{\bullet}\; \tilde{c} = \tilde{b}$, $\tilde{a} \;\tilde{\bullet}\; \tilde{b} = \tilde{a}$ and $\tilde{a} \;\tilde{\bullet}\; \tilde{c} = \tilde{a}$. Then,

$$(\tilde{a} \;\tilde{\bullet}\; \tilde{b}) \;\tilde{\bullet}\; \tilde{c} = \tilde{a} \tag{4.7}$$

$$= \tilde{a} \;\tilde{\bullet}\; (\tilde{b} \;\tilde{\bullet}\; \tilde{c}) \tag{4.8}$$

Case 2: $m_b = 0$ and $m_c = 1$. The binary mask of b if off while that of c is on, so $\tilde{b} \;\tilde{\bullet}\; \tilde{c} = \tilde{b}$, then:

$$(\tilde{a} \;\tilde{\bullet}\; \tilde{b}) \;\tilde{\bullet}\; \tilde{c} = \tilde{a} \;\tilde{\bullet}\; \tilde{b} \tag{4.9}$$

$$= \tilde{a} \;\tilde{\bullet}\; (\tilde{b} \;\tilde{\bullet}\; \tilde{c}) \tag{4.10}$$

Case 3: $m_b = 0$ and $m_c = 0$. No mask is applied, then the MAO is equivalent to the underlying operator $\bullet$, which is associative by Definition 4.

Note the case $m_b = 1$ and $m_c = 0$ violates associativity, but it is impossible under our initial assumption of a right-padding mask sequence $\{m_a, m_b, m_c\}$. □

Given the associativity of MAOs, they can be readily used in parallel scan routines. In practice, augmenting existing SSM and KF operators with their MAO counterpart is a minor code change. MAOs act as a pass-through of the hidden state when padding is applied, thus yielding the correct state at every time step of the padded sequence for each element of the batch without additional indexing or bookkeeping. Due to their pass-through nature, MAOs require strictly equal or less evaluations of the underlying associative operator, which may yield faster runtimes if the operator is expensive to evaluate and/or many elements of the input sequence are masked.

MAOs for SSMs and KFs. As a concrete example, the associative operators for SSMs and KFs involve matrix product and addition. A compute-efficient implementation of MAOs for such operators involves sparse matrix operations, where the sparsity is dictated by the padding mask. However, sparse matrix operations are only expected to yield better runtime than their dense counterparts for large matrices with sufficient levels of sparsity, which are not typical in our application. Thus, no speed-up is expected from using MAOs in the context of this work.

MAOs are similar to the custom operator proposed by Lu et al. (2023), but their effect is fundamentally different: Lu et al. (2023) considers on-policy RL, where the goal is to handle multi-episode sequences, thus their custom operator resets the hidden state at episode boundaries. Instead, in our off-policy RAC architecture, MAOs act as pass-through of the hidden state for padded inputs.

[2] A right-padding mask is a sequence $\{m_0, m_1, \dots\}$ with $m_i \in \{0,1\}$ such that if $m_i = 1$ then $m_j = 1$ for all $j > i$.

4.5. Experiments

In this section, we evaluate the RAC architecture under different history encoders in various POMDPs.

4.5.1. Baselines

We consider the following implementation of history encoders within the RAC architecture.

vSSM. Vanilla, real-valued SSM with diagonal matrices. It is equivalent to a KF layer with infinite observation noise, i.e., the update step has no influence on the output. It can also be seen as a simplification of the S4D model (Gu et al., 2022b), where states are real-valued rather than complex (as in Mega (Ma et al., 2023), such that the recurrence can be interpreted as an exponential moving average) and the recurrence is implemented with a parallel scan rather than a convolution (as in (Smith et al., 2023)).

vSSM+KF. Probabilistic SSM via the KF layers described in Figure 4.2, which is equivalent as adding Kalman filtering on top of vSSM.

vSSM+KF-u. Equivalent to vSSM+KF *without* the input signal $u_{:t}$. It maintains the uncertainty-based gating from the KF layer, but looses flexibility in the KF predict step to influence the prior belief via the input.

Mamba (Gu and Dao, 2023). Selective state-space model with input-dependent state transition matrices.

GRU (Cho et al., 2014). Stateful model with a gating mechanism and non-linear state transitions.

vTransformer (Vaswani et al., 2017). Vanilla encoder-only transformer model with sinusoidal positional encoding and causal self-attention.

All SSM-based approaches are implemented using MAOs and parallel scans. Besides these memory-based agents, we include two additional memoryless agents that implement the same RAC architecure but without embedders or history encoders.

Oracle. It has access to the underlying state of the environment, effectively removing the partial observability aspect of the problem. This method should upper-bound the performance of history encoders.

Memoryless. Unlike Oracle, it does not have access to the underlying state of the environment. This method should lower-bound the performance of history encoders.

All the baselines share a common codebase and hyperparameters. For all stateful models, we use the same latent state dimension N such that parameter count falls within a 10% tolerance range except for GRU, which naturally has more parameters due to its gating mechanism (roughly 40% increase). For vTransformer we choose the dimension of the feed-forward blocks such that the total parameter count is also within 10% of the SSM methods. With this controlled experimental setup, we aim to evaluate strengths and weaknesses of the different mechanisms for sequence modelling (gating, input-selectivity, probabilistic filtering, self-attention) in a wide variety of partially observable environments.

4.5.2. Implementation Details

In this section, we provide details of various components of the RAC architecture and the specific implementations of history encoders. All methods are implemented in a common codebase written in the Pytorch framework (Paszke et al., 2019). Specific hyperparameters used in experiments are included in Appendix C.1.

Embedder. We embed the concatenated observation-action history with a simple linear layer mapping from the combined observation-action dimension to the embedding dimension E.

Soft Actor-Critic. We use a standard SAC implementation with optional automatic entropy tuning (Haarnoja et al., 2019). For discrete action spaces, we use the discrete version of SAC by (Christodoulou, 2019) and one-hot encode the actions.

`vSSM`, `vSSM+KF` & `vSSM+KF-u`. These methods share a similar implementation, with an input linear layer, a linear recurrence and an output linear layer. `vSSM` is equivalent to only using the "Predict" block from the KF layer, while `vSSM+KF-u` removes the input signal $u_{:t}$. For all methods, we discretize the SSM using the zero-order hold method and a learnable scalar step size Δ. In practice we use an auxiliary learnable parameter $\tilde{\Delta}$ and define $\Delta = \texttt{softplus}(\tilde{\Delta})$ to ensure a positive step size. as similarly done in Mamba. We initialize $\tilde{\Delta}$ with a negative value such that after passing through the softplus and after ZOH discretization, the SSM is initialized with eigenvalues close to 1 (i.e., slow decay of state information over time).

`Mamba`. Standard Mamba model from Gu and Dao (2023). We use a reference open-source implementation[3] and modify the parallel scan to use the associated MAO.

`GRU`. Standard implementation included in Pytorch.

`vTransformer`. Default implementation of a causal transformer encoder from Pytorch. We additionally include a sinusoidal positional encoding, as done in prior work using transformers for RL (Ni et al., 2023).

4.5.3. Probabilistic Reasoning - Adaptation and Generalization

We evaluate probabilistic reasoning capabilities with a carefully designed POMDP that simplifies our running example from Section 4.1, where an AI chatbot probes a user in order to recommend a restaurant. Given noisy observations (user's answers) sampled from a bandit with distribution $\mathcal{N}(\mu_b, \sigma_b)$ (user's preference), the task is to infer whether the mean μ_b lies above or below zero (binary decision between two restaurants A and B). At the start of each episode, μ_b and σ_b (the latent parameters) are sampled from some given distribution (i.e., we have a different user every episode). Then, at each step of an episode, the RL agent has three choices: (1) request a new observation from the bandit (ask a new question to the user), which incurs a cost ρ, (2) decide the arm has mean above zero (recommend restaurant A) or (3) decide the arm has mean below zero (recommend restaurant B), both of which immediately end the episode and provide a positive reward if the decision was correct, or a negative reward if the decision was incorrect (i.e., reward is based on whether the recommendation matches the user's preference). We set a maximum episode length of 1000 steps; if the agent does not issue a decision by then, it receives the negative reward. Example rollouts for this environment are provided in Figure 4.3. Given the Bayesian state from Figure 4.3, an optimal agent must strike a balance between requesting new information (which reduces uncertainty about the estimated mean) and minimizing costs. Effective history encoders for this problem should similarly produce a state representation that encodes uncertainty about the latent parameters.

[3]`https://github.com/johnma2006/mamba-minimal/tree/03de542a36d873f6e6c4057ad687278cc6ae944d`

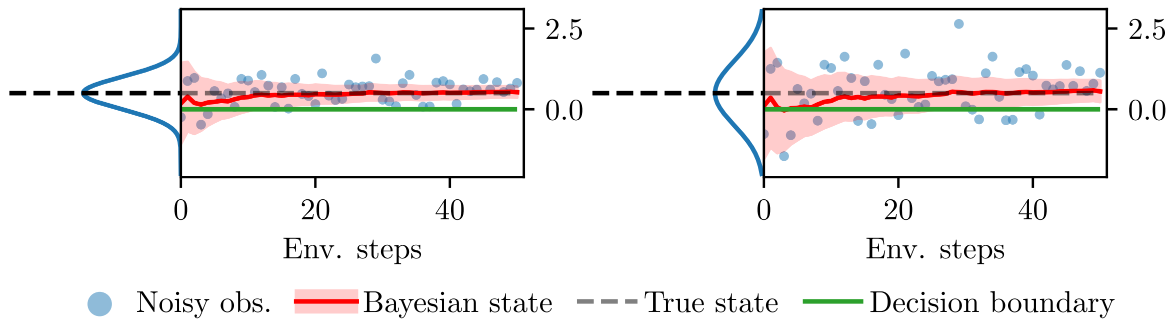


Figure 4.3.: Two example episodes of the Best Arm Identification task of Section 4.5.3, with $\mu_b = 0.5$ and two different noise scales. **(Left)** Narrow noise distribution with $\sigma_b = 0.5$. **(Right)** Wide noise distribution with $\sigma_b = 1.0$. In red, we visualize the Bayesian posterior mean and 3σ confidence interval around μ_b, obtained via Bayesian linear regression using all prior observations in the episode.

We evaluate two core capabilities: adaptation and generalization. Intuitively, an optimal policy for this problem must be *adaptive* depending on the latent parameters. For example, if μ_b is close to zero the agent might need many observations to make an informed decision, whereas with a large $|\mu_b|$ the correct decision can be made with few observations. Moreover, we can also evaluate *generalization* of the learned policy by testing on latent parameters not seen during training. Our hypothesis is that an agent that learns proper probabilistic reasoning (e.g., Bayes' rule) should generalize reasonably well in this task.

We conduct experiments for all baselines under increasing cost ρ. Instead of providing the latent parameters directly to the `Oracle` baseline, we provide the Bayesian posterior mean and standard deviation around the latent parameter μ_b, as shown in Figure 4.3. The agents are trained under the latent parameter distribution given by $\mu_b \sim$ Unif$(-0.5, 0.5)$ and $\sigma_b \sim$ Unif$(0.0, 2.0)$. We additionally evaluate out-of-distribution generalization by using $\sigma_b^{\mathrm{OOD}} \sim$ Unif$(2.0, 3.0)$, i.e., we test how the agent generalizes to bandits with higher variance. In Figure 4.4 we report the normalized return and average episode length for both the training and out-of-distribution latent parameters. Full training curves are included in Appendix C.2. `vSSM+KF` achieves the highest return out of the memory-based agents, both in and out-of-distribution, while matching the performance of `Oracle` in-distribution. The better performance of `vSSM+KF` correlates with longer episodes: compared to the other baselines, `vSSM+KF` learns to request more observations in order to issue a more informed decision.

`vSSM+KF` improves adaptation and generalization. To gain further insights on the results, we do a post-training evaluation on a subset of the agents across the entire latent parameter space, as shown in Figure 4.5. `vSSM+KF` learns adaptation patterns similar to `Oracle`: the length of episodes increase as the noise scale σ_b increases and decrease as $|\mu_b|$ increases, as it is intuitively expected. Such adaptation is less pronounced in `vSSM`, `vSSM+KF-u` and `Mamba`, where episodes are shorter and ultimately results in lower win rates. While `vSSM+KF` does not match the generalization performance of `Oracle`, it remains the best amongst the history encoder baselines. Given our controlled experimental setup, we attribute the enhanced adaptation and generalization of `vSSM+KF` to the internal probabilistic filtering implemented in the KF layer. Moreover, comparing `vSSM+KF` and `vSSM+KF-u` highlights that including the input signal in the KF layer leads to improved performance in this task.

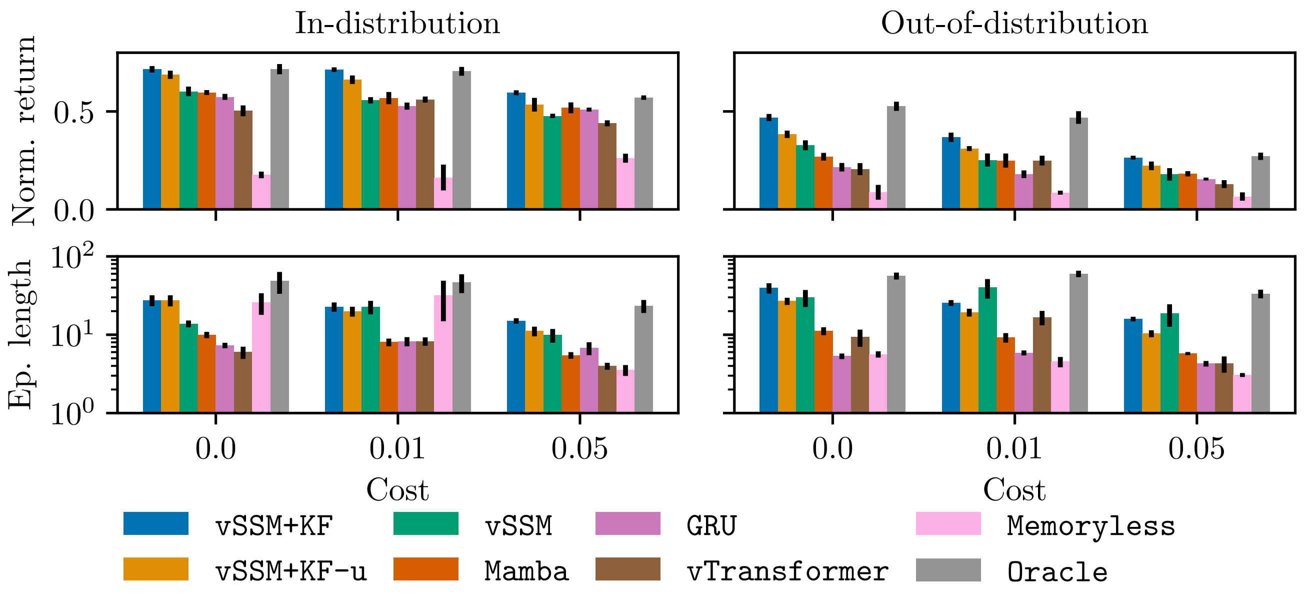


Figure 4.4.: Performance of sequence models in the Best Arm Identification problem after $500\mathrm{K}$ environment steps. We conduct experiments for increasing cost of requesting new observations and evaluate performance both in and out of distribution, averaged over 100 episodes, and report the mean and standard error over 5 random seeds. **(Top row)** Normalized return, obtained by dividing returns by the reward given after winning (10 in our case). **(Bottom row)** Length of episodes.

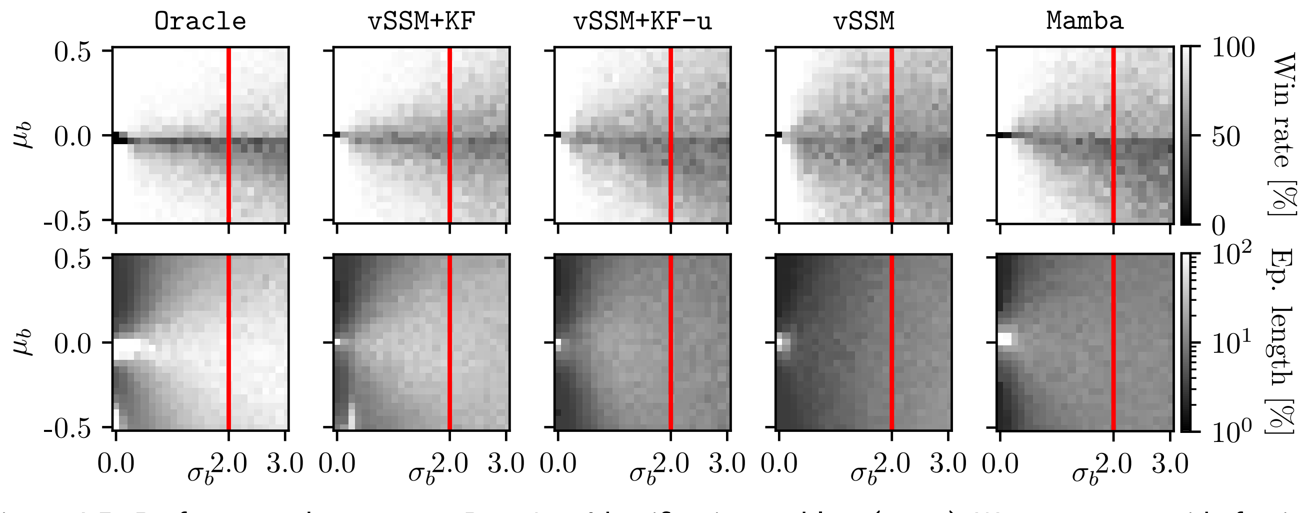


Figure 4.5.: Performance heatmap on Best Arm Identification problem ($\rho = 0$). We generate a grid of noise parameters (μ_b, σ_b) for a total of 625 unique combinations. The red vertical line separates training (to the left) from out-of-distribution (to the right) latent parameters. For each pair of latent parameters, we evaluate performance on five independently trained agents over 100 episodes and report the average win rate and episode lengths.

vSSM+KF can handle adversarial episodes. In Figure 4.6 we compare latent space rollouts[4] from vSSM+KF

[4]We use a latent state dimension $N = 2$ in order to plot the policy decision boundary in latent space. This results in slightly worse performance than the results reported in Figure 4.4, where we use $N = 128$.

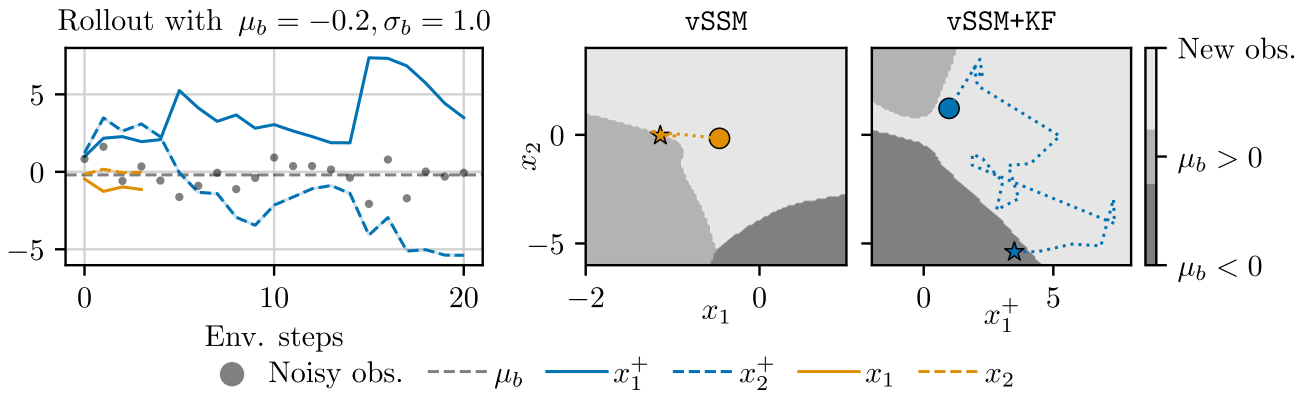


Figure 4.6.: Latent space rollouts in adversarial Best Arm Identification episode. **(Left)** Rollout in latent space ($N = 2$) for vSSM+KF and vSSM after training. **(Middle-Right)** Policy decision boundaries overlaid with the latent space trajectory. Circles and stars denote the beginning and end of trajectories, respectively.

and vSSM in an adversarial episode: μ_b is negative, but the first two observations are positive and of relatively large magnitude. After only four observations, vSSM is mislead by the positive observations and issues the wrong decision, as visualized in Figure 4.6 (middle) where we show the policy's output across latent space, overlaid with the rollout trajectory. Instead, vSSM+KF remains in the region where the policy requests more observations before it navigates to the correct region of latent space, as shown in Figure 4.6 (right). While this example was hand-picked, it is consistent with the adaptation patterns from Figure 4.5.

4.5.4. Probabilistic Filtering - Continuous Control under Observation Noise

In this experiment, we evaluate the ability to learn control policies subject to observation noise. Effective history encoders must learn to aggregate observations over multiple time steps to produce a filtered state representation amenable for control. Our hypothesis is that internal probabilistic filtering provides an inductive bias for learning such a filtered representation. To test our hypothesis, we conduct evaluations across nine environments from the DeepMind Control (DMC) suite (Tunyasuvunakool et al., 2020) with zero-mean Gaussian noise added to the observations, as done by Becker and Neumann (2022); Becker et al. (2024). We present aggregated performance in Figure 4.7 following the recommendations from (Agarwal et al., 2021). We now discuss the main insights from this experiment.

vSSM+KF improves performance of stateful models. The KF layer is the only evaluated add-on for stateful models that significantly improves performance over the baseline model vSSM. This suggests that the uncertainty-based gating in Kalman filters is more effective at handling noisy data compared to the gating mechanism implemented by GRU and Mamba. This observation matches the results in the Best Arm Identification problem from Section 4.5.3. Comparing vSSM+KF and vSSM+KF-u, there is a slight improvement in performance from using an input signal in the KF layer, but it is not statistically significant.

vSSM+KF learns consistently across environments. From the detailed results in Figure 4.8, we observe that vSSM+KF consistently improves performance over the Memoryless lower-bound and achieves the best or comparable final performance in five out of nine tasks. Instead, GRU, Mamba and vTransformer completely fail to learn in some tasks, barely matching the performance of Memoryless.

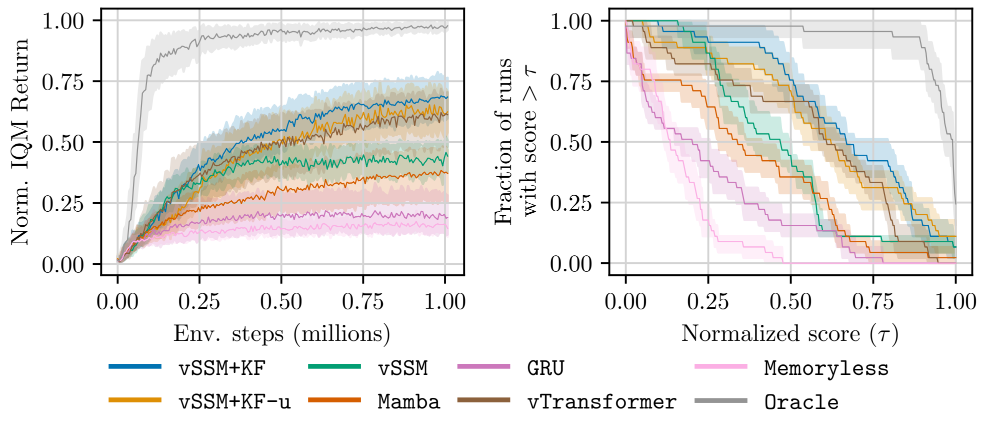


Figure 4.7.: Aggregated performance in noisy DMC benchmark (9 tasks) with 95% bootstrap confidence intervals over five random seeds. **(Left)** Inter-quartile mean returns normalized by the score of `Oracle`. **(Right)** Performance profile after $1\mathrm{M}$ environment steps. Higher curves correspond to better performance.

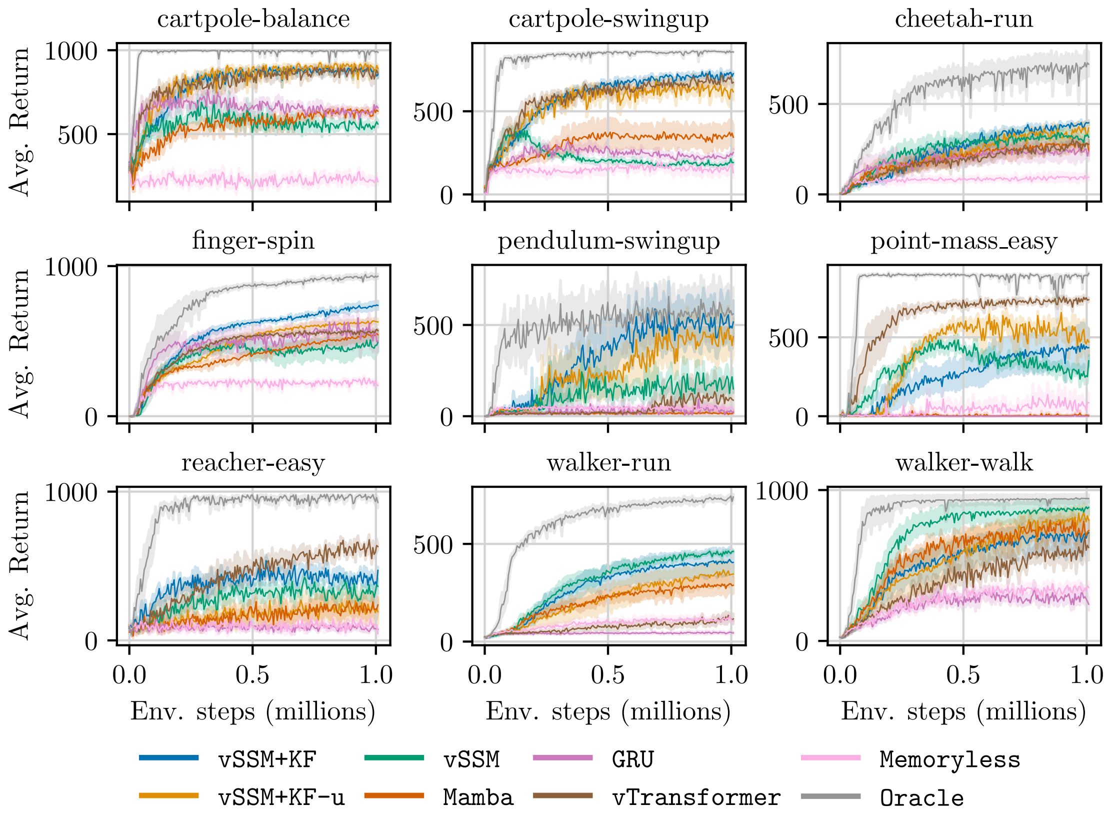


Figure 4.8.: Training curves for the noisy DMC benchmark. We show mean and standard error over five random seeds. For all tasks, we add zero-mean Gaussian noise to the observations with a scale of 0.3, except the `pendulum-swingup` and `point-mass` where the scale is 0.1.

We conduct an additional ablation over increasing noise levels in six representative tasks from the DMC suite, as shown in Figure 4.9. Training curves are included in Appendix C.4

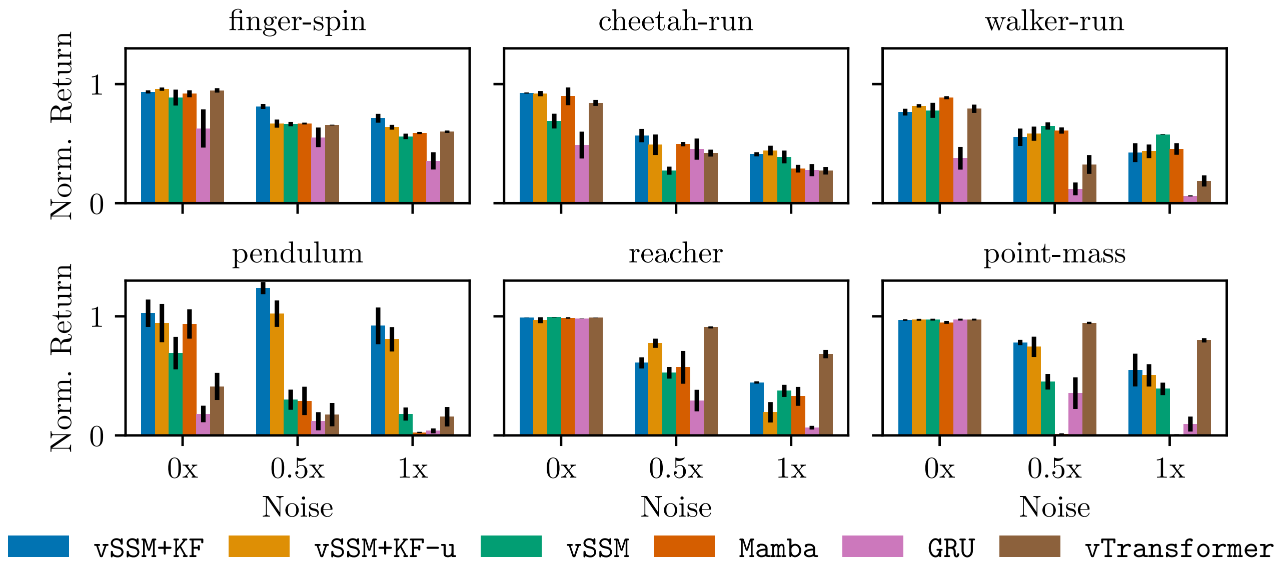


Figure 4.9.: Final performance comparison of recurrent models in six tasks over increasing noise levels. We report the mean and standard error over five random seeds (ten for `pendulum` due to large variance) of the return after 1M environment steps, normalized by the score of `Oracle`.

`vSSM+KF` performs close to `Oracle` under full observability. We observe `vSSM+KF` generally matches the performance of `Oracle` in the absence of noise (normalized score close to 1.0), whereas `vSSM` and `vTransformer` significantly underperform in some tasks. This suggests that the added probabilistic filtering in `vSSM+KF` is a general-purpose strategy even under full observability.

`vSSM+KF`'s robustness to noise is environment dependent. Figure 4.9 suggests that robustness to noise depends generally on the environment, without any clear patterns related to task specifics. `vSSM+KF` is more robust in `finger-spin`, `cheetah-run` and `pendulum`[5], `vSSM` is more robust in `walker-run` but significantly underperforms in other environments, and `vTransformer` is more robuts in `reacher` and `point-mass` but fails to learn in `pendulum`. Overall, `vSSM+KF` shows the most consistent performance across environments and noise levels.

4.5.5. General Memory Capabilities

So far the evaluations were conducted in tasks where probabilistic filtering was intuitively expected to excel. In this experiment, we evaluate performance in a wider variety of POMDPs from the POPGym (Morad et al., 2023) benchmark. We select a subset of 12 tasks that probe models for long-term memory, compression, recall, control under noise and reasoning. The aggregated results are shown in Figure 4.10 and full training curves are also included in Appendix C.5. Below we discuss the main insights.

KF layers can be generally helpful in POMDPs. From the performance profile in Figure 4.10 we observe a statistically significant gap between `vSSM` and `vSSM+KF`. Interestingly, the largest improvements in sample-efficiency (`RepeatPreviousEasy`) and final performance (`MineSweeperEasy`) correspond to

[5]We found that `Oracle` underperforms in the noiseless `pendulum-swingup`, similarly reported in (Luis et al., 2023b), which is why the normalized score in this task is larger than 1.0 in some cases. Moreover, performance does not strictly decrease under higher noise levels, perhaps because noise may actually help avoid early convergence under sparse rewards.

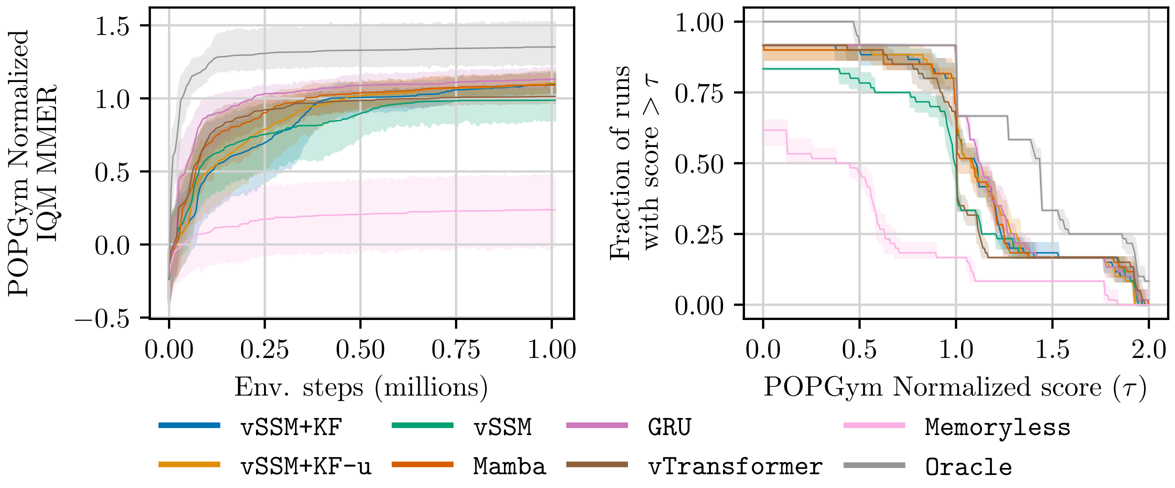


Figure 4.10.: Aggregated performance in POPGym selected environments (12 tasks) with 95% bootstrap confidence intervals over five random seeds. We normalize the maximum-mean episodic return (MMER) by the best reported MMER in (Morad et al., 2023)**(Left)** Normalized IQM MMER **(Right)** Performance profile after $1\mathrm{M}$ environment steps. Higher curves correspond to better performance and a score of 1.0 means equivalent performance as the best baseline (per environment) reported in POPGym.

tasks that probe for memory duration and recall, respectively. The parameter count difference between vSSM and vSSM+KF in these problems is less than 6%, so we believe model capacity is unlikely the reason behind the large performance difference. We hypothesize that, while probabilistic filtering is not required to solve these tasks, the KF layer has extra flexibility via the latent observation and noise signals to accelerate the learning process. We also highlight that vSSM+KF and vSSM+KF-u show comparable performance in this benchmark, suggesting the input signal to be less critical in general memory tasks.

vSSM+KF is less sample-efficient in pure-memory tasks. In particular, we observe that Mamba's input-selectivity is the best-suited mechanism for SSM agents to solve long-term memory problems, matching the performace of GRU and vTransformer. This is an expected result based on the associative recall performance of Mamba reported in its original paper (Gu and Dao, 2023).

Linear SSMs can have strong performance. Morad et al. (2023) report poor performance when combining PPO with the S4D (Gu et al., 2022b) model. While we do not evaluate the S4D model and use an off-policy algorithm in our RAC architecture, our evaluation shows various linear SSMs have strong performance, often surpassing the best reported scores in Morad et al. (2023). Our observation is consistent with the strong performance of PPO with the S5 model reported by Lu et al. (2023).

4.5.6. Ablation

We conduct an ablation on vSSM+KF where we vary two hyperparameters: the latent state size N and the number of stacked KF layers L[6]. We select four representative tasks from POPGym that test different memory capabilities. The final scores are presented in Figure 4.11 and the full training curves are shown in Figure 4.12. Performance is most sensitive to these hyperparameters in the RepeatFirstMedium task,

[6]We use an RMSNorm output block in vSSM+KF since it was critical to ensure stable learning when $L > 1$.

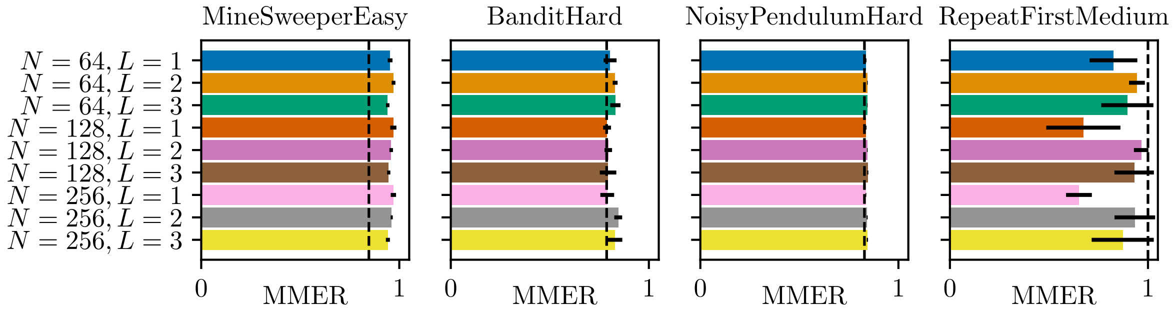


Figure 4.11.: POPGym ablation for vSSM+KF over the latent state size N and the number of layers L. We report the mean and standard error over five random seeds of the MMER score after 1M environment steps. The MMER score is shifted from $[-1, 1]$ to $[0, 1]$ for easier visualization. The vertical line represents the best score reported by Morad et al. (2023).

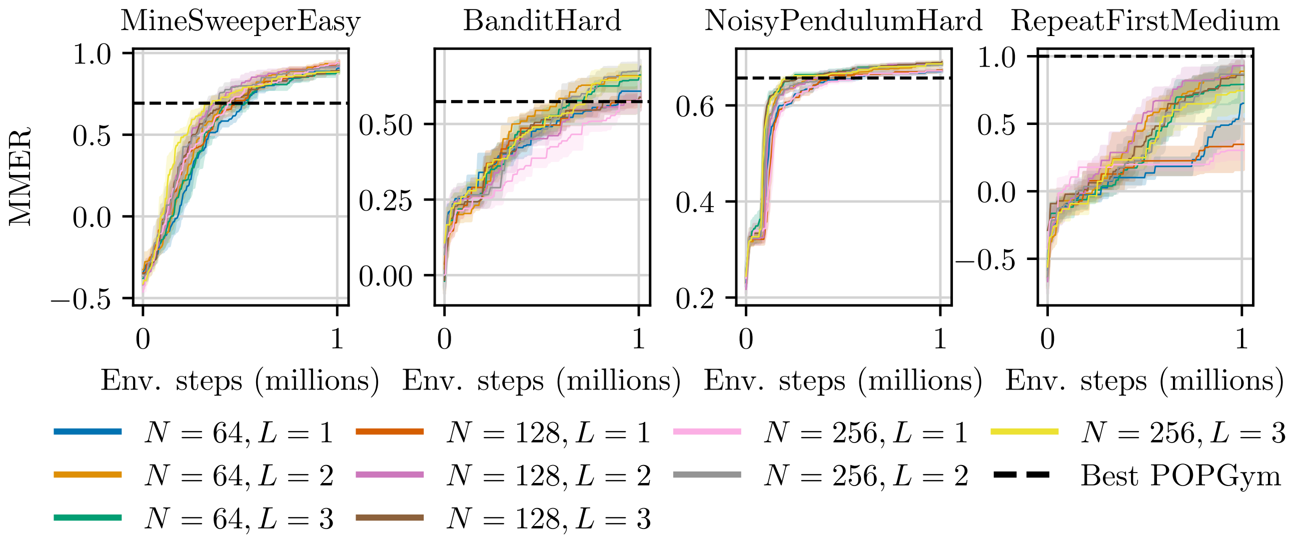


Figure 4.12.: POPGym training curves for ablation experiment over the latent state size N and the number of KF layers L. We show mean and standard error over five random seeds.

where the agent must recall information from the first observation over several steps. The general trend is that using more than one layer improves final performance and increases sample-efficiency (see the training curves in Figure 4.12). Our results are aligned with the good performance of stacked S5 layers reported by Lu et al. (2023), but differ from the observations in (Ni et al., 2023), where both LSTM and transformer models performed best with a single layer in a similar long-term memory task (T-maze passive). From these observations, we believe an interesting avenue for future work is to study what mechanisms enable effective stacking and combination of multiple recurrent layers.

4.6. Conclusion

We investigated the use of Kalman filter (KF) layers as sequence models in a recurrent actor-critic architecture. These layers perform closed-form Gaussian inference in latent space and output a *filtered* state representation for downstream RL components, such as value functions and policies. Thanks to the associative nature of the Kalman filter equations, the KF layers process sequential data efficiently via parallel scans, whose runtime scales logarithmically with the sequence length. To handle trajectories with variable length in off-policy RL, we introduced Masked Associative Operators (MAOs), a general-purpose method that augments any associative operator to recover the correct hidden state when processing padded input data. The KF layers are used as a drop-in replacement for RNNs and SSMs in recurrent architectures, and thus can be trained similarly in an end-to-end, model-free fashion for return maximization.

We evaluated and analysed the strengths and weaknesses of several sequence models in a wide range of POMDPs. KF layers excel in tasks where uncertainty reasoning is key for decision-making, such as the Best Arm Identification task and control under observation noise, significantly improving performance over stateful models like RNNs and deterministic SSMs. In more general tasks, including long-term memory and associative recall, KF layers typically match the performance of transformers and other stateful sequence models, albeit with a lower sample-efficiency.

Limitations and Future Work. We highlight notable limitations of our methodology and suggest avenues for future work. First, we investigated two design decisions in KF layers related to time-varying process noise and posterior covariance as output features. While they resulted in worse performance (see Appendix C.3), in principle they generalize KF layers and may bring benefits in other tasks or contexts, so we believe it is worth further investigation. Second, we use models with relatively low parameter count (< 1M) which is standard in RL but not on other supervised learning tasks. It may be possible that deeper models with larger parameter counts enable new capabilities, e.g., probabilistic reasoning, without explicit probabilistic filtering mechanisms. Third, `vSSM+KF` uses KF layers as standalone history encoders, but more complex architectures may be needed to stabilize training at larger parameter counts. Typical strategies found in models like `Mamba` include residual connections, layer normalization, convolutions and non-linearities. Fourth, our evaluations were limited to POMDPS with relatively low-dimensional observation and action spaces, where small models have enough capacity for learning. Future work could further evaluate performance in more complex POMDPs and compare with our findings.

5. Conclusion

The goal of this thesis was to develop uncertainty estimation techniques to enhance the performance of reinforcement learning algorithms, considering several challenges such as exploration, offline optimization, adaptation and partial observability. In Section 5.1, we discuss how each chapter of this dissertation tackles some of these challenges. In Section 5.2, we describe interesting areas for future research.

5.1. Summary

Chapter 2 addressed the problem of estimating the epistemic variance of value functions in model-based RL. We derived an uncertainty Bellman equation that converges to the variance of values given a posterior distribution of MDPs, thus closing a theoretical gap from prior work and bringing new understanding on how to propagate uncertainty in the decision-making process. Our theoretical framework for uncertainty estimation was paired with optimistic policy optimization to achieve lower regret in challenging exploration problems in tabular RL. Furthermore, we extended our approach to tackle MDPs with continuous state-action spaces leveraging neural networks as function approximators. The resulting algorithm, Q-Uncertainty Soft Actor-Critic (QU-SAC) could flexibly tackle exploration and offline optimization problems with minimal changes. In both problems, quantifying uncertainty around the value function and using it for policy optimization proved instrumental for better performance.

Next, in Chapter 3 we investigated rich distributional representations of epistemic uncertainty around value functions, given parameter uncertainty in MDPs. On the theoretical side, we proved convergence of a so-called value-distributional Bellman operator to the value distribution of a policy. From this theoretical result, we proposed a simple model-based algorithm named Epistemic Quantile Regression (EQR) that iteratively estimates the quantiles of the value distribution. We proposed a deep RL architecture based on EQR, denoted EQR-SAC, that uses neural networks for modelling the quantile representation of the value distribution. The rich representation of uncertainty allows more flexibility in the policy's objective function, which can be specified as any differentiable function of the estimated quantiles. EQR-SAC showed improved sample-efficiency in several continuous control problems and also robustness to the choice of hyperparameters.

Lastly, Chapter 4 considered the problem of partial observability in RL, with a focus in problems where uncertainty about the infered latent state was crucial for decision-making. To tackle these problems, we proposed the use of Kalman filter (KF) layers as drop-in replacement for RNNs and deterministic SSMs in a recurrent actor-critic architecture. Despite their simplified inference scheme, we empirically demonstrated strong performance of KF layers in tasks that required adaptation and probabilistic reasoning capabilities, surpassing commonly used models such as transformers and GRUs. Our work highlighted the strengths of principled probabilistic inference mechanisms for modelling sequences in decision-making.

5.2. Outlook

In this section, we propose and discuss several exciting avenues of future work.

Lifting the theory assumptions. The theory developed in Chapters 2 and 3 rest on the general assumption that the Bayesian posterior around the transition function is independent of the Bayesian posterior around the value function. While this is not restrictive in the tabular case, it is immediately violated when the transition function generalizes across the state-action space, i.e., under function approximation. Given the general importance of function approximation to tackle problems with large (even infinite) state-action spaces, an interesting area of future work is to relax the assumptions in our theory of uncertainty-related Bellman operators. For instance, by extending the family of MDPs (e.g., linear) where the epistemic uncertainty around the value function still admits a recursive formulation akin the UBE or the value-distributional Bellman equation. While under relaxed assumptions these Bellman equations may no longer exist, we believe future work should still aim for uncertainty estimation techniques that inherently consider the decision-making nature of RL, as those would naturally perform propagation of uncertainty.

Model learning. Accurate and well-calibrated dynamic models are essential in model-based RL (Curi et al., 2020; Malik et al., 2019), however it was not a focus in this dissertation. We believe that advances in dynamic models will naturally translate to improved uncertainty quantification in value functions. First, while ensembles (Lakshminarayanan et al., 2017) have favorable statistical and practical properties, in this work we used relatively small ensembles (< 10 members) for computational efficiency, which may limit their uncertainty expressivity. Future work could further explore the effect of various ensembling techniques in the estimated value uncertainty. Second, in this work we generally used short model rollouts (<50 steps) to avoid compunding errors (Lambert et al., 2022), but longer rollouts are desirable to capture the long-term effect of policies in a given dynamical system. Third, another interesting area of future work is to leverage Bayesian neural networks as the backbone for learning environment dynamics (Depeweg, 2019).

Distributional representations. In Chapter 3 we used the quantile parameterization from (Dabney et al., 2018b), which is widely adopted (Wurman et al., 2022) and easy to implement. However, thanks to the close relationship between our Bayesian perspective and the model-free distributional RL literature, we believe that advances in the latter can be equally leveraged by the former. From theoretical analysis (Rowland et al., 2018, 2023) to novel distributional representations (Dabney et al., 2018a; Yang et al., 2019), we argue that drawing strong connections between (epistemic) value distributions and (aleatoric) return distributions may lead to interesting insights.

Joint objectives. A weakness of the uncertainty-aware framework introduced in Chapter 1 is that it treats model learning, value uncertainty and policy optimization as three separate learning problems: dynamics are trained with a supervised loss to accurately predict next states, value functions and their uncertainty estimates are trained to minimize some Bellman error and policies are trained to maximize some value-based objectives. The potential mismatch between two or more of these objectives is often labelled as a weakness of model-based approaches, e.g., more accurate models may not yield better policies (Lambert et al., 2020). Future work could investigate how to best integrate joint model-policy objective functions, such as the one proposed by Eysenbach et al. (2022), with the uncertainty quantification methods developed in this dissertation.

Scaling probabilistic inference in SSMs. In Chapter 4 we introduced a simple KF layer as a mechanism for probabilistic inference in RL under partial observability. Since the KF layer is a standalone layer, it can be combined with other layers and components to build complex architectures. We empirically analyzed relatively simple architectures at low parameter counts, with few stacked KF layers and an output normalization block. However, sequence models in other domains such as language and vision require careful architectural decisions and training recipes in order to train much deeper models with large parameter counts, which enable good performance and generalization in complex tasks. It remains an open question how to best scale up sequence models with internal probabilistic inference mechanisms and whether new capabilities emerge at larger parameter counts.

Combining gating mechanisms in recurrent models. In Chapter 4 we highlighted that KF layers implement an uncertainty-based gating mechanism, while other sequence models like Mamba (Gu and Dao, 2023) and GRUs (Cho et al., 2014) implement other gating mechanisms. We empirically analyze the strengths and weaknesses of such gating mechanisms, but an equally interesting research question is how to best *combine* these techniques into a single gating mechanism. For instance, combining the uncertainty-based gating of KF layers with the input-dependent gating of Mamba could yield models that are equally performant at probabilistic reasoning and long-term memory tasks.

Explicit usage of latent state uncertainty. The recurrent actor-critic architecture proposed in Chapter 4 only actively uses the *filtered* state representation from KF layers to condition the actor and the critic. However, actively using the *uncertainty* of the latent state for decision-making is an open research problem. Our initial experiments using the posterior variance of the latent state as an output feature of KF layers yielded worse performance than only using the posterior mean (see results in Appendix C.3). Nevertheless, we believe explicit usage of the uncertainty around the infered latent state is required for generally-capable agents under partial observability.

A. Supplementary Material for Chapter 2

A.1. Theory Proofs

A.1.1. Proof of Theorem 1

In this section, we provide the formal proof of Theorem 1. We begin by showing an expression for the posterior variance of the value function without assumptions on the MDP. We define the joint distribution $p^\pi(a, s' \mid s) = \pi(a \mid s)p(s' \mid s, a)$ for a generic transition function p. To ease notation, since π is fixed, we will simply denote the joint distribution as $p(a, s' \mid s)$.

Lemma 2. *For any $s \in \mathcal{S}$ and any policy π, it holds that*

$$\mathbb{V}_P\left[V^{\pi(s)}\right] = \gamma^2\,\mathbb{E}_P\left[\left(\sum_{a,s'} P(a, s' \mid s)V^\pi(s')\right)^2\right] - \gamma^2\left(\mathbb{E}_P\left[\sum_{a,s'} P(a, s' \mid s)V^\pi(s')\right]\right)^2. \tag{A.1}$$

Proof. Using the Bellman expectation equation

$$v^\pi(s) = \sum_a \pi(a \mid s)r(s, a) + \gamma \sum_{a,s'} p(a, s' \mid s)v^\pi(s'), \tag{A.2}$$

we have

$$\mathbb{V}_P\left[V^\pi(s)\right] = \mathbb{V}_P\left[\sum_a \pi(a \mid s)r(s, a) + \gamma \sum_{a,s'} P(a, s' \mid s)V^\pi(s')\right] \tag{A.3}$$

$$= \mathbb{V}_P\left[\gamma \sum_{a,s'} P(a, s' \mid s)V^\pi(s')\right], \tag{A.4}$$

where (A.4) holds since $r(s, a)$ is deterministic. Using the identity $\mathbb{V}[Y] = \mathbb{E}[Y^2] - (\mathbb{E}[Y])^2$ on (A.4) concludes the proof. □

The next result is the direct consequence of our set of assumptions.

Lemma 3. *Under Assumptions 1–3, $P(s' \mid s, a)$ and $V^\pi(s')$ are conditionally independent random variables for all triplets $(s, a, s') \in \mathcal{S} \times \mathcal{A} \times \mathcal{S}$.*

Proof. Let $T_{0:\infty}$ be a random trajectory under the random transition dynamics P. Under Assumptions 2 and 3, $T_{0:\infty}$ is a sequence of H random, but *unique* states followed by the terminal (absorbing) state $\{S_0, S_1, \dots, S_H, s_T, s_T, \dots\}$, i.e., we have $S_i \neq S_j$ for all $i \neq j$. We prove the Lemma by dividing the random trajectory into two segments: the random (finite) segment for step $h \leq H$ and the deterministic (infinite) segment for $h > H$.

Case $T_{0:H}$. Under Assumption 1, the conditioned trajectory probability $\mathbb{P}(T_{0:H} \mid P)$, which is itself a random variable through conditioning on P, is a product of independent random variables defined by

$$\mathbb{P}(T_{0:H} \mid P) = \prod_{h=0}^{H-1} \pi(A_h \mid S_h) P(S_{h+1} \mid S_h, A_h) \tag{A.5}$$

$$= P(S_1 \mid S_0, A_0)\pi(A_0 \mid S_0) \prod_{h=1}^{H-1} \pi(A_h \mid S_h) P(S_{h+1} \mid S_h, A_h). \tag{A.6}$$

$$= P(S_1 \mid S_0, A_0)\pi(A_0 \mid S_0)\, \mathbb{P}(T_{1:H} \mid P). \tag{A.7}$$

Note that each transition probability in $\mathbb{P}(T_{0:H} \mid P)$ is distinct by Assumption 2. Additionally, for any $h > 0$ the action probability $\pi(A_h \mid S_h)$ is *conditionally independent* of $P(S_h \mid S_{h-1}, A_{h-1})$ given S_h. Then, for arbitrary $S_0 = s$, $A_0 = a$ and $S_1 = s'$, we have that $P(s' \mid s, a)$ is conditionally independent of $\mathbb{P}(T_{1:H-1} \mid P)$. Since $V^\pi(S_1 \mid S_1 = s')$ is a function of $\mathbb{P}(T_{1:H} \mid P)$, then it is also conditionally independent of $P(s' \mid s, a)$.

Case $T_{H:\infty}$. The Lemma is trivially satisfied since both the transition probability and the values become constants: we have $P(s_T \mid s_T, a) = 1$ and $V^\pi(s_T) = 0$.

Combining both results, we have that P and V^π are conditionally independent for any arbitrary infinite trajectory, which completes the proof. ☐

Using the previous result yields the following lemma.

Lemma 4. *Under Assumptions 1–3, it holds that*

$$\sum_{a,s'} \mathbb{E}_P\big[P(a, s' \mid s) V^\pi(s')\big] = \sum_{a,s'} \bar{p}(a, s' \mid s)\, \mathbb{E}_P\big[V^\pi(s')\big]. \tag{A.8}$$

Proof. For any pair of random variables X and Y on the same probability space, by definition of covariance it holds that $\mathbb{E}[XY] = \mathrm{Cov}[X, Y] + \mathbb{E}[X]\,\mathbb{E}[Y]$. Using this identity, the fact that the independence result from Lemma 3 implies zero correlation and the definition of posterior mean transition in (2.3) yields the result. ☐

Now we are ready to prove the main theorem.

Theorem 1. *Under Assumptions 1–3, for any $s \in \mathcal{S}$ and policy π, the posterior variance of the value function, $U^\pi = \mathbb{V}_P[V^\pi]$ obeys the uncertainty Bellman equation*

$$U^\pi(s) = \gamma^2 u(s) + \gamma^2 \sum_{a,s'} \pi(a \mid s) \bar{p}(s' \mid s, a) U^\pi(s'), \tag{2.7}$$

where $u(s)$ is the local uncertainty defined as

$$u(s) = \mathbb{V}_{a,s' \sim \pi, \bar{p}}\big[\bar{v}^\pi(s')\big] - \mathbb{E}_P\Big[\mathbb{V}_{a,s' \sim \pi, P}\big[V^\pi(s')\big]\Big]. \tag{2.8}$$

Proof. Starting from the result in Lemma 2, we consider each term on the r.h.s of (A.1) separately. For the first term, notice that within the expectation we have a squared expectation over the transition probability $P(s' \mid s,a)$, thus using the identity $(\mathbb{E}[Y])^2 = \mathbb{E}[Y^2] - \mathbb{V}[Y]$ results in

$$\mathbb{E}_P\left[\left(\sum_{a,s'} P(a,s' \mid s)V^\pi(s')\right)^2\right] = \mathbb{E}_P\left[\sum_{a,s'} P(a,s' \mid s)\big(V^\pi(s')\big)^2 - \mathbb{V}_{a,s'\sim\pi,P}\big[V^\pi(s')\big]\right]. \tag{A.9}$$

Applying linearity of expectation to bring it inside the sum and an application of Lemma 4 (note that the lemma applies for squared values as well) gives

$$= \sum_{a,s'} \bar{p}(a,s' \mid s)\,\mathbb{E}_P\Big[\big(V^\pi(s')\big)^2\Big] - \mathbb{E}_P\Big[\mathbb{V}_{a,s'\sim\pi,P}\big[V^\pi(s')\big]\Big]. \tag{A.10}$$

For the second term of the r.h.s of (A.1) we apply again Lemma 4 and under definition of variance

$$\left(\mathbb{E}_P\left[\sum_{a,s'} P(a,s' \mid s)V^\pi(s')\right]\right)^2 = \left(\sum_{a,s'} \bar{p}(a,s' \mid s)\,\mathbb{E}_P\big[V^\pi(s')\big]\right)^2 \tag{A.11}$$

$$= \sum_{a,s'} \bar{p}(a,s' \mid s)\Big(\mathbb{E}_P\big[V^\pi(s')\big]\Big)^2 - \mathbb{V}_{a,s'\sim\pi,\bar{p}}\Big[\mathbb{E}_P\big[V^\pi(s')\big]\Big]. \tag{A.12}$$

Finally, since

$$\mathbb{E}_P\Big[\big(V^\pi(s')\big)^2\Big] - \Big(\mathbb{E}_P\big[V^\pi(s')\big]\Big)^2 = \mathbb{V}_P\big[V^\pi(s')\big] \tag{A.13}$$

for any $s' \in \mathcal{S}$, we can plug (A.10) and (A.12) into (A.1), which proves the theorem. □

A.1.2. Proof of Theorem 2

In this section, we provide the supporting theory and the proof of Theorem 2. First, we will use the identity $\mathbb{V}[\mathbb{E}[Y|X]] = \mathbb{E}[(\mathbb{E}[Y|X])^2] - (\mathbb{E}[E[Y|X]])^2$ to prove $u(s) = w(s) - g(s)$ holds, with $Y = \sum_{a,s'} P(a,s' \mid s)V^\pi(s')$. For the conditioning variable X, we define a transition function with fixed input state s as a mapping $p_s : \mathcal{A} \to \Delta(S)$ representing a distribution $p_s(s' \mid a) = p(s' \mid s,a)$. Then $X = \mathbf{P}_s := \big\{P_s(s' \mid a)\big\}_{s'\in\mathcal{S},a\in\mathcal{A}}$. The transition function P_s is drawn from a distribution Φ_s obtained by marginalizing Φ on all transitions not starting from s.

Lemma 5. *Under Assumptions 1–3, it holds that*

$$\mathbb{V}_{P_s}\left[\mathbb{E}_P\left[\sum_{a,s'} P(a,s' \mid s)V^\pi(s') \,\middle|\, \mathbf{P}_s\right]\right] = \mathbb{V}_P\left[\sum_{a,s'} P(a,s' \mid s)\bar{v}^\pi(s')\right]. \tag{A.14}$$

Proof. Treating the inner expectation,

$$\mathbb{E}_P\left[\sum_{a,s'} P(a,s' \mid s)V^\pi(s') \mid \mathbf{P}_s\right] = \sum_a \pi(a \mid s)\sum_{s'} \mathbb{E}_P\big[P(s' \mid s,a)V^\pi(s') \mid \mathbf{P}_s\big]. \tag{A.15}$$

Due to the conditioning, $P(s' \mid s, a)$ is deterministic within the expectation

$$= \sum_{a,s'} P(a, s' \mid s)\, \mathbb{E}_P\big[V^\pi(s') \mid \mathbf{P}_s\big]. \tag{A.16}$$

By Lemma 3, $V^\pi(s')$ is independent of $\mathbf{P}_s$, so we can drop the conditioning

$$= \sum_{a,s'} P(a, s' \mid s)\bar{v}^\pi(s'). \tag{A.17}$$

Lastly, since drawing samples from a marginal distribution is equivalent to drawing samples from the joint, i.e., $\mathbb{V}_X[f(X)] = \mathbb{V}_{(X,Y)}[f(X)]$, then:

$$\mathbb{V}_{P_s}\left[\sum_{a,s'} P(a, s' \mid s)\bar{v}^\pi(s')\right] = \mathbb{V}_P\left[\sum_{a,s'} P(a, s' \mid s)\bar{v}^\pi(s')\right], \tag{A.18}$$

completing the proof. □

The next lemma establishes the result for the expression $\mathbb{E}[(\mathbb{E}[Y|X])^2]$.

Lemma 6. *Under Assumptions 1–3, it holds that*

$$\mathbb{E}_{P_s}\left[\left(\mathbb{E}_P\left[\sum_{a,s'} P(a, s' \mid s)V^\pi(s') \,\middle|\, \mathbf{P}_s\right]\right)^2\right] = \sum_{a,s'} \bar{p}(a, s' \mid s)\big(\bar{v}^\pi(s')\big) - \mathbb{E}_P\Big[\mathbb{V}_{a,s'\sim\pi,P}\big[\bar{v}^\pi(s')\big]\Big]. \tag{A.19}$$

Proof. The inner expectation is equal to the one in Lemma 5, so we have that

$$\left(\mathbb{E}_P\left[\sum_{a,s'} P(a, s' \mid s)V^\pi(s') \,\middle|\, \mathbf{P}_s\right]\right)^2 = \left(\sum_{a,s'} P(a, s' \mid s)\bar{v}^\pi(s')\right)^2 \tag{A.20}$$

$$= \sum_{a,s'} P(a, s' \mid s)\big(\bar{v}^\pi(s')\big)^2 - \mathbb{V}_{a,s'\sim\pi,P}\big[\bar{v}^\pi(s')\big]. \tag{A.21}$$

Finally, applying expectation on both sides of (A.21) yields the result. □

Similarly, the next lemma establishes the result for the expression $(\mathbb{E}[\mathbb{E}[Y|X]])^2$.

Lemma 7. *Under Assumptions 1–3, it holds that*

$$\left(\mathbb{E}_{P_s}\left[\mathbb{E}_P\left[\sum_{a,s'} P(a, s' \mid s)V^\pi(s') \,\middle|\, \mathbf{P}_s\right]\right]^2\right) = \sum_{a,s'} \bar{p}(a, s' \mid s)\big(\bar{v}^\pi(s')\big) - \mathbb{V}_{a,s'\sim\pi,\bar{p}}\big[\bar{v}^\pi(s')\big]. \tag{A.22}$$

Proof. By the tower property of expectations, $(\mathbb{E}[\mathbb{E}[Y|X]])^2 = (\mathbb{E}[Y])^2$. Then, the result follows directly from (A.11) and (A.12). □

The second part of Theorem 2 is a corollary of the next lemma.

Lemma 8. *Under Assumptions 1 and 2, it holds that*

$$\mathbb{E}_P\Big[\mathbb{V}_{a,s'\sim\pi,P}\big[V^\pi(s')\big] - \mathbb{V}_{a,s'\sim\pi,P}\big[\bar{v}^\pi(s')\big]\Big] \tag{A.23}$$

is non-negative.

Proof. We will prove the lemma by showing (A.23) is equal to $\mathbb{E}_P\Big[\mathbb{V}_{a,s'\sim\pi}\big[V^\pi(s') - \bar{v}^\pi(s')\big]\Big]$, which is a non-negative quantiy by definition of variance. The idea is to derive two expressions for $\mathbb{E}[\mathbb{V}[Y|X]]$ and compare them. First, we will use the identity $\mathbb{E}[\mathbb{V}[Y|X]] = \mathbb{E}[\mathbb{E}[(Y - \mathbb{E}[Y|X])^2|X]]$. The outer expectation is w.r.t the marginal distribution Φ_s while the inner expectations are w.r.t Φ. For the inner expectation we have

$$\mathbb{E}_P\left[\left(\sum_{a,s'} P(a,s' \mid s)V^\pi(s') - \mathbb{E}_P\left[\sum_{a,s'} P(a,s' \mid s)V^\pi(s') \,\middle|\, \mathbf{P}_s\right]\right)^2 \,\middle|\, \mathbf{P}_s\right] \tag{A.24}$$

$$= \mathbb{E}_P\left[\left(\sum_{a,s'} P(a,s' \mid s)\big(V^\pi(s') - \mathbb{E}_P[V^\pi \mid \mathbf{P}_s]\big)\right)^2 \,\middle|\, \mathbf{P}_s\right] \tag{A.25}$$

$$= \mathbb{E}_P\left[\left(\sum_{a,s'} p(a,s' \mid s)\big(V^\pi(s') - \bar{v}^\pi(s')\big)\right)^2 \,\middle|\, \mathbf{P}_s\right] \tag{A.26}$$

$$= \mathbb{E}_P\left[\sum_{a,s'} P(a,s' \mid s)\big(V^\pi(s') - \bar{v}^\pi(s')\big)^2 - \mathbb{V}_{a,s'\sim\pi,P}\big[V^\pi(s') - \bar{v}^\pi(s')\big] \,\middle|\, \mathbf{P}_s\right] \tag{A.27}$$

$$= \sum_{a,s'} P(a,s' \mid s)\,\mathbb{V}_P\big[V^\pi(s')\big] - \mathbb{E}_P\Big[\mathbb{V}_{a,s'\sim\pi,P}\big[V^\pi(s') - \bar{v}^\pi(s')\big] \,\Big|\, \mathbf{P}_s\Big]. \tag{A.28}$$

Applying the outer expectation to the last equation, along with Lemma 3 and the tower property of expectations yields:

$$\mathbb{E}[\mathbb{V}[Y|X]] = \sum_{a,s'} \bar{p}(a,s' \mid s)\,\mathbb{V}_P\big[V^\pi(s')\big] - \mathbb{E}_P\Big[\mathbb{V}_{a,s'\sim\pi,P}\big[V^\pi(s') - \bar{v}^\pi(s')\big]\Big]. \tag{A.29}$$

Now we repeat the derivation but using $\mathbb{E}[\mathbb{V}[Y|X]] = \mathbb{E}[\mathbb{E}[Y^2|X] - (\mathbb{E}[Y|X])^2]$. For the inner expectation of the first term we have:

$$\mathbb{E}_P\left[\left(\sum_{a,s'} P(a,s' \mid s)V^\pi(s')\right)^2 \,\middle|\, \mathbf{P}_s\right] \tag{A.30}$$

$$= \mathbb{E}_P\left[\sum_{a,s'} P(a,s' \mid s)\big(V^\pi(s')\big)^2 - \mathbb{V}_{a,s'\sim\pi,P}\big[V^\pi(s')\big] \,\middle|\, \mathbf{P}_s\right]. \tag{A.31}$$

Applying the outer expectation:

$$\mathbb{E}[\mathbb{E}[Y^2|X]] = \sum_{a,s'} \bar{p}(a,s' \mid s)\, \mathbb{E}_P\Big[\big(V^\pi(s')\big)^2\Big] - \mathbb{E}_P\Big[\mathbb{V}_{a,s'\sim\pi,P}\big[V^\pi(s')\big]\Big]. \tag{A.32}$$

Lastly, for the inner expectation of $\mathbb{E}[(\mathbb{E}[Y|X])^2]$:

$$\left(\mathbb{E}_P\left[\sum_{a,s'} P(a,s' \mid s) V^\pi(s') \,\middle|\, \mathbf{P}_s\right]\right)^2 = \left(\sum_{a,s'} P(a,s' \mid s)\bar{v}^\pi(s')\right)^2 \tag{A.33}$$

$$= \sum_{a,s'} P(a,s' \mid s)\big(\bar{v}^\pi(s')\big)^2 - \mathbb{V}_{a,s'\sim\pi,P}\big[\bar{v}^\pi(s')\big]. \tag{A.34}$$

Applying the outer expectation:

$$\mathbb{E}[(\mathbb{E}[Y|X])^2] = \sum_{a,s'} \bar{p}(a,s' \mid s)\big(\bar{v}^\pi(s')\big)^2 - \mathbb{E}_P\Big[\mathbb{V}_{a,s'\sim\pi,P}\big[\bar{v}^\pi(s')\big]\Big]. \tag{A.35}$$

Finally, by properties of variance, (A.29) = (A.32) - (A.35) which gives the desired result. □

Theorem 2. *Under Assumptions 1–3, for any $s \in \mathcal{S}$ and policy π, it holds that $u(s) = w(s) - g(s)$, where $g(s) = \mathbb{E}_P\Big[\mathbb{V}_{a,s'\sim\pi,P}\big[V^\pi(s')\big] - \mathbb{V}_{a,s'\sim\pi,P}\big[\bar{v}^\pi(s')\big]\Big]$. Furthermore, we have that the gap $g(s)$ is non-negative, thus $u(s) \le w(s)$.*

Proof. By definition of $u(s)$ in (2.8), proving the claim is equivalent to showing

$$\mathbb{V}_{a,s'\sim\pi,\bar{p}}\big[\bar{v}^\pi(s')\big] = w(s) + \mathbb{E}_{p\sim\Phi}\Big[\mathbb{V}_{a,s'\sim\pi,P}\big[\bar{v}^\pi(s')\big]\Big], \tag{A.36}$$

which holds by combining Lemmas 5–7. Lastly, $u(s) \le w(s)$ holds by Lemma 8. □

A.2. Theory Extensions

A.2.1. Unknown Reward Function

We can easily extend the derivations on Appendix A.1.1 to include the additional uncertainty coming from an *unknown* reward function. Similarly, we model the reward function as a random variable R drawn from a prior distribution $\Psi(R)$, and whose belief will be updated via Bayes rule. In this new setting, we now consider the variance of the values under the distribution of MDPs, represented by the random variable $\mathcal{M}$. We need the following additional assumptions to extend our theory.

Assumption 4 (Independent rewards)**.** *$R(x,a)$ and $R(y,a)$ are independent random variables if $x \neq y$.*

Assumption 5 (Independent transitions and rewards)**.** *The random variables $P(\cdot \mid s,a)$ and $R(s,a)$ are independent for any (s,a).*

With Assumption 4 we have that the value function of next states is independent of the transition function and reward function at the current state. Assumption 5 means that sampling $\mathcal{M} \sim \Gamma$ is equivalent as independently sampling $P \sim \Phi$ and $R \sim \Psi$.

Theorem 4. *Under Assumptions 1–5, for any $s \in \mathcal{S}$ and policy π, the posterior variance of the value function, $U^\pi = \mathbb{V}_{\mathcal{M}}[V^\pi]$ obeys the uncertainty Bellman equation*

$$U^\pi(s) = \mathbb{V}_R\left[\sum_a \pi(a \mid s) R(s,a)\right] + \gamma^2 u(s) + \gamma^2 \sum_{a,s'} \pi(a \mid s)\bar{p}(s' \mid s,a) U^\pi(s'), \tag{A.37}$$

where $u(s)$ is defined in (2.8).

Proof. By Assumptions 4 and 5 and following the derivation of Lemma 2 we have

$$\mathbb{V}_{\mathcal{M}}\big[V^\pi(s)\big] = \mathbb{V}_{\mathcal{M}}\left[\sum_a \pi(a \mid s)R(s,a) + \gamma \sum_{a,s'} P(a,s' \mid s) V^\pi(s')\right] \tag{A.38}$$

$$= \mathbb{V}_R\left[\sum_a \pi(a \mid s) R(s,a)\right] + \mathbb{V}_{\mathcal{M}}\left[\gamma \sum_{a,s'} P(a,s' \mid s) V^\pi(s')\right]. \tag{A.39}$$

Then following the same derivations as Appendix A.1.1 completes the proof. □

A.2.2. Extension to Q-values

Our theoretical results naturally extend to action-value functions. The following result is analogous to Theorem 1.

Theorem 5. *Under Assumptions 1–3, for any $(s,a) \in \mathcal{S} \times \mathcal{A}$ and policy π, the posterior variance of the Q-function, $U^\pi = \mathbb{V}_P[Q^\pi]$ obeys the uncertainty Bellman equation*

$$U^\pi(s,a) = \gamma^2 u(s,a) + \gamma^2 \sum_{a',s'} \pi(a' \mid s')\bar{p}(s' \mid s,a) U^\pi(s',a'), \tag{A.40}$$

where $u(s,a)$ is the local uncertainty defined as

$$u(s,a) = \mathbb{V}_{a',s' \sim \pi, \bar{p}}\big[\bar{q}^\pi(s',a')\big] - \mathbb{E}_{p \sim \Phi}\Big[\mathbb{V}_{a',s' \sim \pi, P}\big[Q^\pi(s',a')\big]\Big] \tag{A.41}$$

Proof. Follows the same derivation as Appendix A.1.1 □

Similarly, we can connect to the upper-bound found by Zhou et al. (2020) with the following theorem.

Theorem 6. *Under Assumptions 1–3, for any $(s,a) \in \mathcal{S} \times \mathcal{A}$ and policy π, it holds that $u(s,a) = w(s,a) - g(s,a)$, where $w(s,a) = \mathbb{V}_P\Big[\sum_{a',s'} \pi(a' \mid s')P(s' \mid s,a)\bar{q}^\pi(s',a')\Big]$ and $g(s,a) = \mathbb{E}_P\Big[\mathbb{V}_{a',s' \sim \pi, P}\big[Q^\pi(s',a')\big] - \mathbb{V}_{a',s' \sim \pi, P}\big[\bar{q}^\pi(s',a')\big]\Big]$. Furthermore, we have that the gap $g(s,a) \geq 0$ is non-negative, thus $u(s,a) \leq w(s,a)$.*

Proof. Follows the same derivation as Appendix A.1.2. Similarly, we can prove that the gap $g(s,a)$ is non-negative by showing it is equal to $\mathbb{E}_P\Big[\mathbb{V}_{a',s' \sim \pi, P}\big[Q^\pi(s',a') - \bar{q}^\pi(s',a')\big]\Big]$. □

A.2.3. State-Action Uncertainty Rewards

In our practical experiments, we use the results of both Appendices A.2.1 and A.2.2 to compose the uncertainty rewards propagated via the UBE. Concretely, we consider the following two approaches for computing state-action uncertainty rewards:

- `pombu`:

$$w(s, a) = \mathbb{V}_P \left[\sum_{a', s'} \pi(a' \mid s') P(s' \mid s, a) \bar{q}^{\pi}(s', a') \right] \tag{A.42}$$

- `exact-ube`:

$$u(s, a) = w(s, a) - \mathbb{E}_P \Big[\mathbb{V}_{a', s' \sim \pi, P} \big[Q^{\pi}(s', a') - \bar{q}^{\pi}(s', a') \big] \Big] \tag{A.43}$$

Additionally, since we also learn the reward function, we add to the above the uncertainty term generated by the reward function posterior, as shown in Appendix A.2.1: $\mathbb{V}_R \big[R(s, a) \big]$.

A.3. Hyperparameters

Table A.1.: Hyperparameters for the DMC experiments of Section 2.7.3. For MBPO, we use of $M = 2$ as the original method uses clipped Q-learning.

Name	Value
General	
T - # episodes	500
E - steps per episode	10^3
Replay buffer $\mathcal{D}$ capacity	10^5
Batch size (all nets)	256
Warm-up steps (under initial policy)	5×10^3
SAC	
G - # gradient steps	10
Auto-tuning of entropy coefficient α?	Yes
Target entropy	$-\dim(\mathcal{A})$
Actor MLP network	2 hidden layers - 128 neurons - Tanh activations
Critic MLP network	2 hidden layers - 256 neurons - Tanh activations
Actor/Critic learning rate	3×10^{-4}
Dynamics Model	
N - ensemble size	5
F - frequency of model training (# steps)	250
L - # model rollouts per step	400
k - rollout length	5
Δ - # Model updates to retain data	1
Model buffer(s) capacity	$L \times F \times k \times \Delta = 5 \times 10^5$
Model MLP network	4 layers - 200 neurons - SiLU activations
Learning rate	1×10^{-3}
QU-SAC Specific	
M - # critics per dynamics model	1
λ - # uncertainty gain	1.0
Uncertainty type	$\{$`ensemble-var`, `upper-bound`$\}$

Table A.2.: Hyperparameters for the D4RL experiments of Section 2.7.4.

Name	**Value**
General	
G - gradient steps	10^6
Replay buffer $\mathcal{D}$ capacity	10^6
Batch size (all nets)	512
SAC	
Auto-tuning of entropy coefficient α?	Yes
Target entropy	$-\dim(\mathcal{A})$
Actor MLP network	3 hidden layers - 256 neurons - Tanh activations
Critic MLP network	3 hidden layers - 256 neurons - Tanh activations
Actor learning rate	3×10^{-5}
Critic learning rate	3×10^{-4}
Dynamics Model	
N - ensemble size	5
F - frequency of data collection (# steps)	1000
L - rollout batch size	5×10^4
k - rollout length	15
Δ - # Data collection calls to retain data	5
Model buffer(s) capacity	$L \times k \times \Delta = 3.75 \times 10^6$
Model MLP network	4 layers - 200 neurons - SiLU activations
Learning rate	1×10^{-3}
QU-SAC Specific	
M - # critics per dynamics model	$\{1, 2\}$
λ - # uncertainty gain	-1.0
Uncertainty type	$\{$`ensemble-var`, `upper-bound`$\}$

B. Supplementary Material for Chapter 3

B.1. Theory Proofs

Proposition 1 (Random Variable Value-Distribution Bellman Equation). *Let V^π be the random value function defined in* (2.2)*. Then, it holds that*

$$V^\pi(s) = \sum_a \pi(a \mid s) r(s,a) + \gamma \sum_{s',a} \pi(a \mid s) P(s' \mid s,a) V^\pi(s'), \tag{3.4}$$

for any policy π and initial state $s \in \mathcal{S}$.

Proof. We proceed similarly as the standard Bellman equation proof shown by Bellemare et al. (2023). First, the random trajectories $\tilde{T}$ have two properties endowed by the Markov decision process: time homogeneity and the Markov property. Informally speaking, time homogeneity states that the trajectory from a given state s is independent of the time k at which the state is visited, while the Markov property states that trajectories starting from s are independent of states, actions or rewards encountered before s (c.f. Bellemare et al. (2023) Lemmas 2.13, 2.14 for a formal definition). In the domain of random variables, these properties imply that two trajectories starting from the same initial state s are equally distributed regardless of past history.

From the definition (2.2) we decompose the random value into the immediate reward and the value at the next state:

$$V^\pi(s) = \mathbb{E}_{\tilde{T}}[R_0 | S_0 = s,\, P] + \gamma\, \mathbb{E}_{\tilde{T}}\left[\sum_{h=1}^{\infty} \gamma^{h-1} R_h \middle| S_0 = s,\, P \right]. \tag{B.1}$$

For the first term, the only random variable remaining is A_0, so we rewrite it as

$$= \sum_a \pi(a \mid s) r(s,a) + \gamma\, \mathbb{E}_{\tilde{T}}\left[\sum_{h=1}^{\infty} \gamma^{h-1} R_h \middle| S_0 = s,\, P \right]. \tag{B.2}$$

For the second term, we apply the tower property of expectations

$$= \sum_a \pi(a \mid s) r(s,a) + \gamma\, \mathbb{E}_{\tilde{T}}\left[\mathbb{E}_{\tilde{T}}\left[\sum_{h=1}^{\infty} \gamma^{h-1} R_h \middle| S_0 = s,\, A_0,\, S_1,\, P \right] \middle| S_0 = s,\, P \right]. \tag{B.3}$$

By the Markov property,

$$= \sum_a \pi(a \mid s) r(s,a) + \gamma\, \mathbb{E}_{\tilde{T}}\left[\mathbb{E}_{\tilde{T}}\left[\sum_{h=1}^{\infty} \gamma^{h-1} R_h \middle| S_1,\, P \right] \middle| S_0 = s,\, P \right]. \tag{B.4}$$

By time homogeneity, the inner expectation is exactly equal to the random variable $V^\pi(S_1)$, after a change of variable in the infinite sum index

$$= \sum_a \pi(a \mid s) r(s,a) + \gamma\, \mathbb{E}_{\tilde{T}}\big[V^\pi(S_1) \big| S_0 = s,\, P \big]. \tag{B.5}$$

Lastly, the remaining random variable is S_1, for which we can explicitly write its probability distribution, concluding the proof

$$= \sum_a \pi(a \mid s) r(s,a) + \gamma \sum_{a,s'} \pi(a \mid s) P(s' \mid s,a) V^\pi(s'). \tag{B.6}$$

□

Notation: for the following Lemma, we use the notation $\mathfrak{D}(X)$ to denote the distribution of the random variable $X \in \mathcal{X}$, as done in Bellemare et al. (2023). In particular, we use $\mathfrak{D}_P$ to denote the distribution of a random variable belonging to the probability space of P, i.e., random variables derived from the posterior distribution $\Phi(P \mid \mathcal{D})$.

Lemma 1 (Value-Distribution Bellman Equation)**.** *The value distribution function μ^π obeys the Bellman equation.*

$$\mu^\pi = \mathbb{E}_P\big[(b_{r^\pi, P^\pi, \gamma})_\# \mu^\pi \big] \tag{3.7}$$

for any policy π.

Proof. In matrix-vector format, the random-variable value-distributional Bellman equation is expressed as

$$\mathbf{V}^\pi = \mathbf{r}^\pi + \gamma \mathbf{P}^\pi \mathbf{V}^\pi. \tag{B.7}$$

Since μ^π refers to the distribution of the random variable $\mathbf{V}^\pi$, which belongs to the probability space of the random transition function P, then we use our notation to write $\mu^\pi = \mathfrak{D}_P(\mathbf{V}^\pi)$. Further, using the notation $\mathfrak{D}_P(\cdot)$ on the r.h.s of (B.7) yields

$$\mu^\pi = \mathfrak{D}_P(\mathbf{r}^\pi + \gamma \mathbf{P}^\pi \mathbf{V}^\pi). \tag{B.8}$$

For any two random variables X, Y in the same probability space, it holds that the marginal distribution over X can be written as the expected value over Y of the conditional distribution. That is, $\mathfrak{D}(X) = \mathbb{E}_Y\big[\mathfrak{D}(X \mid Y)\big]$, which follows from standard probability theory (Wasserman, 2013). Applying this property to the r.h.s of (B.8) results in

$$\mu^\pi = \mathbb{E}_P\big[\mathfrak{D}_P(\mathbf{r}^\pi + \gamma \mathbf{P}^\pi \mathbf{V}^\pi \mid \mathbf{P}^\pi) \big]. \tag{B.9}$$

A similar derivation can be found in related prior work studying the variance of μ^π (O'Donoghue et al., 2018; Zhou et al., 2019; Luis et al., 2023a).

Given that $P(s' \mid s, a)$ and $V^\pi(s')$ are independent under our assumptions, then conditioning on $\mathbf{P}^\pi$ means that the distribution of the matrix-vector product $\mathbf{P}^\pi\mathbf{V}^\pi$ is simply the distribution of applying a linear transformation on $\mathbf{V}^\pi$. The result is that the conditional distribution can be interpreted as the pushforward

$$\mathfrak{D}_P(\mathbf{r}^\pi + \gamma\mathbf{P}^\pi\mathbf{V}^\pi \mid \mathbf{P}^\pi) = (b_{r^\pi, P^\pi, \gamma})_{\#}\mu^\pi, \tag{B.10}$$

which completes the proof. $\square$

We adopt the supremum p-Wasserstein distance to establish contractivity of the operator $\mathcal{T}^\pi$.

Definition 5. For $p \in [1, \infty)$, the p-Wasserstein distance between two distributions ν, ν' is a metric $w_p : \mathcal{P}(\mathbb{R}) \times \mathcal{P}(\mathbb{R}) \to [0, \infty]$ defined by

$$w_p(\nu, \nu') = \left(\int_0^1 \left| F_\nu^{-1}(\tau) - F_{\nu'}^{-1}(\tau) \right|^p d\tau \right)^{1/p}, \tag{B.11}$$

where $F_{(\cdot)}^{-1}$ is the inverse cumulative distribution function. Furthermore, the supremum p-Wasserstein distance $\bar{w}_p$ between two value distribution functions $\mu, \mu' \in \mathcal{P}(\mathbb{R})^{\mathcal{S}}$ is defined by $\bar{w}_p(\mu, \mu') = \sup_{s \in \mathcal{S}} w_p(\mu(s), \mu'(s))$.

The supremum p-Wasserstein distance was proven to be a metric in $\mathcal{P}(\mathbb{R})^{\mathcal{S}}$ by Bellemare et al. (2017).

To prove that $\mathcal{T}^\pi$ is a contraction, we adopt the technique from Bellemare et al. (2023) that relies on the alternative definition of the p-Wasserstein distance in terms of couplings.

Definition 6 (Coupling (Villani, 2008) Definition 1.1 (adapted))**.** Let $\nu, \nu' \in \mathcal{P}(\mathbb{R})$ be two probability distributions over the reals. A coupling between ν and ν' is a joint probability distribution $\upsilon \in \mathcal{P}(\mathbb{R}^2)$ whose marginals are ν and ν'. That is, given random variables $(V, V') \sim \upsilon$, we have $V \sim \nu$ and $V' \sim \nu'$. Further, we denote $\Gamma(\nu, \nu') \subseteq \mathcal{P}(\mathbb{R}^2)$ the set[1] of all couplings between ν and ν'.

Intuitively, the coupling υ can be interpreted as a transport plan to move probability mass from one distribution to another. The p-Wasserstein distance can also be defined as the cost of the optimal transport plan

$$w_p(\nu, \nu') = \min_{\upsilon \in \Gamma(\nu, \nu')} \mathbb{E}_{(V, V') \sim \upsilon}\left[|V - V|^p \right]^{1/p}. \tag{B.12}$$

The existence of an optimal coupling $\upsilon^\star$ that minimizes (B.12) is guaranteed since ν, ν' are measures defined on a complete, separable metric space ($\mathbb{R}$ with the usual metric) and equipped with the corresponding Borel σ-algebra (i.e., ν, ν' are measures on a Polish space) (cf. Villani, 2008, Theorem 4.1).

With these definitions, we now proceed to prove the contraction of the Bellman operator.

Theorem 3. *The operator $\mathcal{T}^\pi$ is a γ-contraction with respect to $\bar{w}_p$ for all $p \in [1, \infty)$. That is, $\bar{w}_p(\mathcal{T}^\pi\mu, \mathcal{T}^\pi\mu') \leq \gamma\bar{w}_p(\mu, \mu')$ for all $\mu, \mu' \in \mathcal{P}(\mathbb{R})^{\mathcal{S}}$ such that $V(s') \sim \mu(s')$, $V'(s') \sim \mu'(s')$ are conditionally independent of $P^\pi(s' \mid s)$ given $s' \in \mathcal{S}$.*

[1]This set is non-empty: there exists a trivial coupling in which the variables V, V' are independent (Villani, 2008)

Proof. We follow closely the proofs of Proposition 4.1 by Amortila et al. (2020) and Proposition 4.15 by Bellemare et al. (2023). For each $s \in \mathcal{S}$, let $\upsilon^\star$ denote the optimal coupling that minimizes the p-Wasserstein metric from Definition 5 between some arbitrary pair of value distributions $\mu(s), \mu'(s) \in \mathcal{P}(\mathbb{R})$, so that $(V(s), V'(s)) \sim \upsilon^\star$.

Define new random variables $\tilde{V}(s) = r^\pi(s) + \gamma \sum_{s'} P^\pi(s' \mid s) V(s')$, $\tilde{V}'(s) = r^\pi(s) + \gamma \sum_{s'} P^\pi(s' \mid s) V'(s')$. By definition of the operator $\mathcal{T}^\pi$, we have that $\tilde{V}(s) \sim (\mathcal{T}^\pi \mu)(s)$ and $\tilde{V}'(s) \sim (\mathcal{T}^\pi \mu')(s)$, which means that the pair $(\tilde{V}(s), \tilde{V}'(s)) \sim \tilde{\upsilon}$ is a coupling between $(\mathcal{T}^\pi \mu)(s)$ and $(\mathcal{T}^\pi \mu')(s)$.

Starting from Definition 6 and since the p-Wasserstein distance is a minimum over couplings, then

$$w_p^p\big((\mathcal{T}\mu)(s), (\mathcal{T}\mu')(s)\big) \leq \mathbb{E}_P\left[\left|\tilde{V}(s) - \tilde{V}'(s)\right|^p\right]. \tag{B.13}$$

Plugging the definition of the random variables,

$$= \mathbb{E}_P\left[\left| r^\pi(s) + \gamma \sum_{s'} P^\pi(s' \mid s) V(s') - r^\pi(s) - \gamma \sum_{s'} P^\pi(s' \mid s) V'(s') \right|^p\right] \tag{B.14}$$

By re-arrangement of terms

$$= \gamma^p \, \mathbb{E}_P\left[\left| \sum_{s'} P^\pi(s' \mid s)(V(s') - V'(s')) \right|^p\right]. \tag{B.15}$$

Since $f(x) = |x|^p$ is convex for $p \geq 1$, then by Jensen's inequality

$$\leq \gamma^p \, \mathbb{E}_P\left[\sum_{s'} P^\pi(s' \mid s) \big|(V(s') - V'(s'))\big|^p \right]. \tag{B.16}$$

By linearity of expectation

$$= \gamma^p \sum_{s'} \mathbb{E}_P\Big[P^\pi(s' \mid s) \big|(V(s') - V'(s'))\big|^p \Big]. \tag{B.17}$$

By the independence assumption on $P^\pi(s' \mid s)$, the expectation of the product becomes the product of expectations

$$= \gamma^p \sum_{s'} \mathbb{E}_P\big[P^\pi(s' \mid s)\big] \, \mathbb{E}_P\Big[\big|(V(s') - V'(s'))\big|^p \Big]. \tag{B.18}$$

Since the supremum of non-negative values is greater or equal than any convex combination of them

$$\leq \gamma^p \sup_{s'} \mathbb{E}_P\Big[\big|(V(s') - V'(s'))\big|^p \Big]. \tag{B.19}$$

By definition of the supremum p-Wasserstein distance

$$= \gamma^p \bar{w}_p^p(\mu, \mu'). \tag{B.20}$$

Taking supremum on the left-hand side and taking the p-th root on both sides completes the proof. □

Theorem 3 parallels similar results in standard RL and model-free distributional RL, in that it allows us to establish the convergence of iterated applications of $\mathcal{T}^\pi$ (Corollary 2).

B.1.1. Supporting Lemmas

Lemma 9. *Define an arbitrary $\mu_0 \in \mathcal{P}_B(\mathbb{R})^{\mathcal{S}}$. Then, under Assumptions 1–3, the sequence $\{\mu_k\}_{k=0}^{\infty}$ defined by $\mu_{k+1} = \mathcal{T}^{\pi}\mu_k$ is such that for all $k \geq 0$ the random variable $V_k(s') \sim \mu_k(s')$ is conditionally independent of $P^{\pi}(s' \mid s)$ given $s' \in \mathcal{S}$.*

Proof. Intuitively, by definition of the operator $\mathcal{T}^{\pi}$, the random variable $V_k(s')$ is the result of summing rewards starting from state s', following the random dynamics P^{π} for k steps and then bootstraping with V_0. That is, applying $\mathcal{T}^{\pi}$ k times results in the random variable

$$V_k(s') = \mathbb{E}_{T_{0:k}}\left[\sum_{h=0}^{k-1} \gamma^h R_h + \gamma^k V_0(S_k) \middle| S_0 = s', P\right], \tag{B.21}$$

where $T_{0:k}$ is a random state trajectory $\{S_0, S_1, ..., S_k\}$ starting from $S_0 = s'$ and following the random dynamics P. Under Assumptions 1–3, $T_{0:k}$ is a sequence of unique states (with the exception of the absorbing terminal state), which means that the probability of returning to s' are zero. Finally, since $V_k(s')$ is an expectation conditioned on the initial state being s', it follows that $V_k(s')$ is conditionally independent of $P^{\pi}(s' \mid s)$ given s'. □

Lemma 10. *If the value distribution function μ has bounded support, then $\mathcal{T}^{\pi}\mu$ also has bounded support.*

Proof. From bounded rewards on $[r_{\min}, r_{\max}]$, then we denote by $\mathcal{P}_B(\mathbb{R})^{\mathcal{S}}$ the space of value distributions bounded on $[v_{\min}, v_{\max}]$, where $v_{\min} = r_{\min}/(1-\gamma)$ and $v_{\max} = r_{\max}/(1-\gamma)$.

Given arbitrary $\mu \in \mathcal{P}_B(\mathbb{R})^{\mathcal{S}}$, let $v(s)$ be a realization of $\mu(s)$ for any $s \in \mathcal{S}$. Then, $\sum_a \pi(a \mid s) r(s,a) + \gamma \sum_{a,s'} \pi(a \mid s) P(s' \mid s,a) v(s')$ is an instantiation of $(\mathcal{T}^{\pi}\mu)(s)$ for any $s \in \mathcal{S}$. We have:

$$\mathbb{P}\big((\mathcal{T}^{\pi}\mu)(s) \leq v_{\max}\big) = \mathbb{P}\left(\sum_a \pi(a \mid s) r(s,a) + \gamma \sum_{a,s'} \pi(a \mid s) P(s' \mid s,a) v(s') \leq v_{\max}\right), \tag{B.22}$$

$$= \mathbb{P}\left(\gamma \sum_{a,s'} \pi(a \mid s) P(s' \mid s,a) v(s') \leq v_{\max} - \sum_a \pi(a \mid s) r(s,a)\right). \tag{B.23}$$

Since $\sum_a \pi(a \mid s) r(s,a) \leq r_{\max}$, then

$$\geq \mathbb{P}\left(\gamma \sum_{a,s'} \pi(a \mid s) P(s' \mid s,a) v(s') \leq v_{\max} - r_{\max}\right). \tag{B.24}$$

By definition of $v_{\max}$

$$\geq \mathbb{P}\left(\sum_{a,s'} \pi(a \mid s) P(s' \mid s,a) v(s') \leq v_{\max}\right). \tag{B.25}$$

Finally, since $v(s') \leq v_{\max}$ for any $s' \in \mathcal{S}$, then

$$= 1. \tag{B.26}$$

Under the same logic, we can similarly show that $\mathbb{P}\big((\mathcal{T}^{\pi}\mu)(s) \geq v_{\min}\big) = 1$, such that $\mathbb{P}\big((\mathcal{T}^{\pi}\mu)(s) \in [v_{\min}, v_{\max}]\big) = 1$ for any $s \in \mathcal{S}$. □

B.2. Hyperparameters

Table B.1.: Hyperparameters for DeepMind Control Suite. In red, we highlight the only deviations of the base hyperparameters across all environments and baselines.

Name	Value
General	
T - # episodes	250
E - steps per episode	10^3
Replay buffer $\mathcal{D}$ capacity	10^5
Warm-up steps (under initial policy)	5×10^3
SAC	
G - # gradient steps	10
Batch size	256
Auto-tuning of entropy coefficient α?	Yes
Target entropy	$-\dim(\mathcal{A})$
Actor MLP network	2 hidden layers - 128 neurons - Tanh activations
Critic MLP network	2 hidden layers - 256 neurons - Tanh activations
Actor/Critic learning rate	3×10^{-4}
Dynamics Model	
n - ensemble size	5
F - frequency of model training (# steps)	250
L - # model rollouts per step	400
k - rollout length	5
Δ - # Model updates to retain data	10
Model buffer $\mathcal{D}_{\text{model}}$ capacity (EQR-SAC)	$L \times F \times k \times \Delta (\times n) = 5 \times 10^6 (25 \times 10^6)$
Model MLP network (quadruped)	4 layers - 200 (400) neurons - SiLU activations
Learning rate	1×10^{-3}
Quantile Network	
m - # quantiles	51
# (s', a') samples (EQR-SAC only)	25

B.3. DM Control Learning Curves

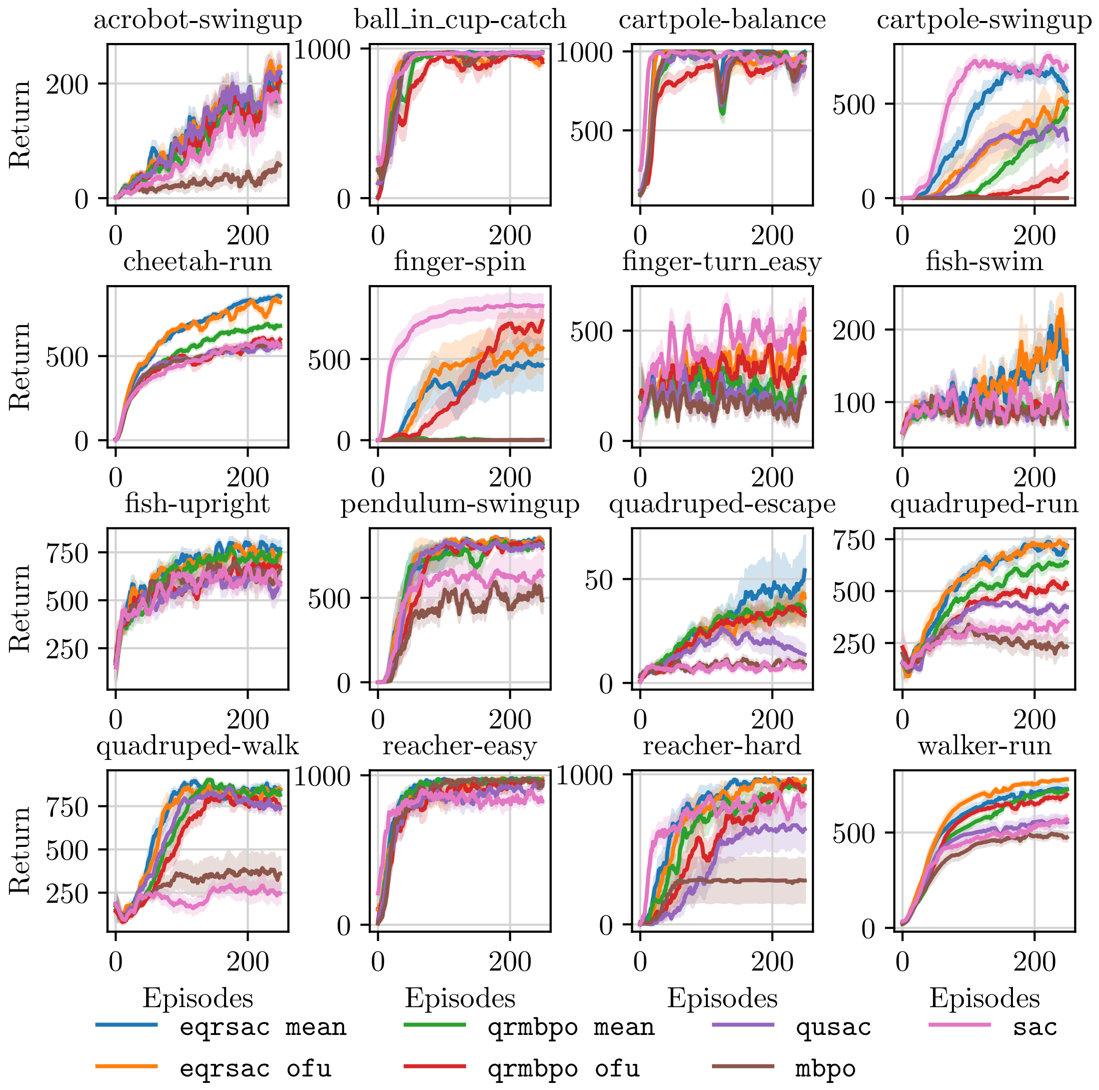


Figure B.1.: Individual learning curves of DMC benchmark.

B.4. DM Control Final Scores

Table B.2.: Scores in DMC benchmark after 250 episodes (or 250K environment steps). For each environment, we report the mean and standard error scores over 10 random seeds. We **bold** the highest final mean score per environment.

Environment	`sac`	`mbpo`	`qrmbpo mean`	`qrmbpo ofu`	`qusac`	`eqrsac mean`	`eqrsac ofu`
acrobot-swingup	167.7 ± 22.4	57.7 ± 19.9	202.3 ± 18.7	204.0 ± 21.4	220.3 ± 30.8	217.8 ± 18.5	$\mathbf{229.4} \pm 17.4$
ball-in-cup-catch	972.6 ± 3.3	972.5 ± 1.7	974.4 ± 2.3	908.3 ± 25.1	972.8 ± 2.8	$\mathbf{977.2} \pm 1.4$	928.7 ± 18.5
cartpole-balance-sparse	949.5 ± 18.7	985.6 ± 9.7	977.0 ± 22.2	894.9 ± 42.8	904.5 ± 37.3	$\mathbf{997.8} \pm 2.1$	968.0 ± 15.2
cartpole-swingup-sparse	$\mathbf{693.8} \pm 27.2$	0.1 ± 0.1	476.2 ± 72.6	131.8 ± 72.1	310.6 ± 64.3	566.4 ± 54.4	510.6 ± 86.9
cheetah-run	551.6 ± 22.6	571.4 ± 19.3	679.1 ± 10.7	598.4 ± 16.2	567.4 ± 16.8	$\mathbf{854.0} \pm 11.7$	820.0 ± 18.4
finger-spin	$\mathbf{827.5} \pm 68.1$	0.1 ± 0.1	1.2 ± 1.1	734.4 ± 89.7	3.2 ± 2.6	567.1 ± 146.7	461.9 ± 154.1
finger-turn-easy	$\mathbf{571.3} \pm 31.3$	220.0 ± 20.0	289.8 ± 34.1	399.0 ± 35.7	220.0 ± 20.0	221.3 ± 19.9	460.5 ± 58.3
fish-swim	79.9 ± 10.9	80.6 ± 10.0	70.0 ± 8.7	91.9 ± 13.3	83.8 ± 10.0	145.1 ± 27.3	$\mathbf{168.3} \pm 20.4$
fish-upright	579.4 ± 50.8	660.5 ± 59.5	749.9 ± 29.1	671.4 ± 32.2	591.9 ± 53.5	$\mathbf{766.2} \pm 45.3$	735.0 ± 23.5
pendulum-swingup	631.0 ± 111.7	484.5 ± 76.4	819.2 ± 17.2	796.9 ± 25.2	808.6 ± 19.7	$\mathbf{834.4} \pm 15.7$	833.7 ± 16.0
quadruped-escape	8.0 ± 1.2	8.8 ± 1.8	34.5 ± 7.1	32.4 ± 5.0	13.7 ± 4.3	$\mathbf{54.2} \pm 16.7$	41.1 ± 11.4
quadruped-run	352.0 ± 36.4	232.3 ± 41.2	638.5 ± 26.0	532.6 ± 19.0	421.9 ± 16.8	$\mathbf{719.8} \pm 19.0$	712.5 ± 23.6
quadruped-walk	245.5 ± 57.4	360.1 ± 95.1	815.1 ± 25.4	739.4 ± 38.2	734.8 ± 26.5	844.3 ± 26.8	$\mathbf{849.2} \pm 17.1$
reacher-easy	824.6 ± 21.9	474.3 ± 20.8	968.6 ± 9.8	959.1 ± 13.2	943.1 ± 13.8	931.2 ± 21.5	$\mathbf{977.9} \pm 2.5$
reacher-hard	797.5 ± 38.8	291.9 ± 146.3	921.8 ± 22.2	905.0 ± 31.6	635.0 ± 139.5	919.6 ± 15.7	$\mathbf{965.3} \pm 9.8$
walker-run	568.9 ± 19.1	474.3 ± 20.8	725.5 ± 10.8	698.9 ± 13.7	553.8 ± 30.9	727.4 ± 24.3	$\mathbf{779.3} \pm 7.9$

C. Supplementary Material for Chapter 4

C.1. Hyperparameters

Table C.1.: Hyperparameters used for Section 4.5. For the `Mamba` parameters, we use the notation from the code by Gu and Dao (2023) and select parameters to match a effective state size $N = 128$. `GRU` and `vTransformer` use default parameters from Pytorch unless noted otherwise.

Parameter	BestArm	DMC	POPGym
Training			
Buffer size	∞		
Adam learning rate	3e-4		
Env. steps	500K	1M	
Batch size	64	32	
Update-to-data (UTD) ratio	0.25	1.0	
# Eval episodes	100	16	
RAC			
Embedding size (E)	16		
Latent size (N)	128		
Activations	ReLU		
Context length	256	64	
Actor MLP	[128]	[256, 256]	
Critic MLP	[256]	[256, 256]	
SAC			
Discount factor γ	0.99		
Entropy temp. α	0.1	Auto	
Target entropy (continuous)	N/A	-dim($\mathcal{A}$)	
Target entropy (discrete)	N/A		$-0.7 \log(1/\text{dim}(\mathcal{A}))^1$
History Encoders (common)			
Latent size N	128		
# layers	1		
`vSSM`, `vSSM+KF` & `vSSM+KF-u`			
$\tilde{\Delta}$ init	-7		
$\mathbf{A}$ init	HiPPO (diagonal)		
$\mathbf{B}$ init	$\mathbf{I}$		
$\mathbf{\Sigma}^{\text{P}}$ init	$\mathbf{I}$		
Inital state belief	$\mathcal{N}(\mathbf{0}, \mathbf{I})$		
RMSNorm output?	No	Yes	No
`Mamba`			
$\mathbf{A}$ init	HiPPO (diagonal)		
`d_model` (embedding size)	16		
`d_state` (per-channel hidden size)	4		
Expand factor E	2		
Size of Δ projection	1		
1D Conv kernel size	4		
`vTransformer`			
# heads	1		
Feedforward size	128	256	

C.2. Best Arm Identification Training Curves

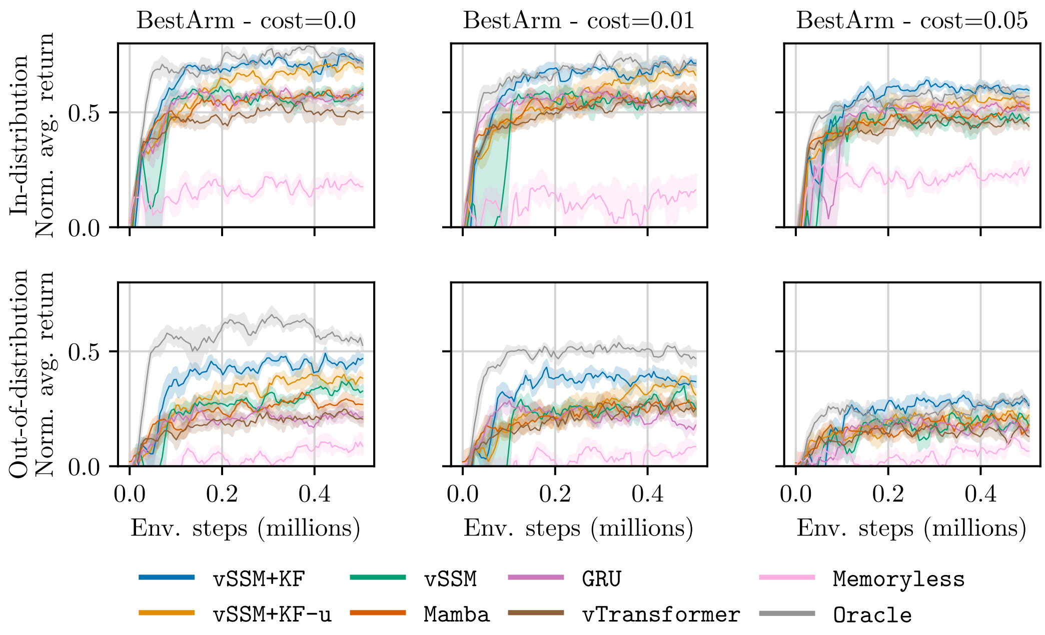


Figure C.1.: Normalized average return over 100 episodes in and out of distribution, for increasing costs. We report the mean and standard error over 5 random seeds.

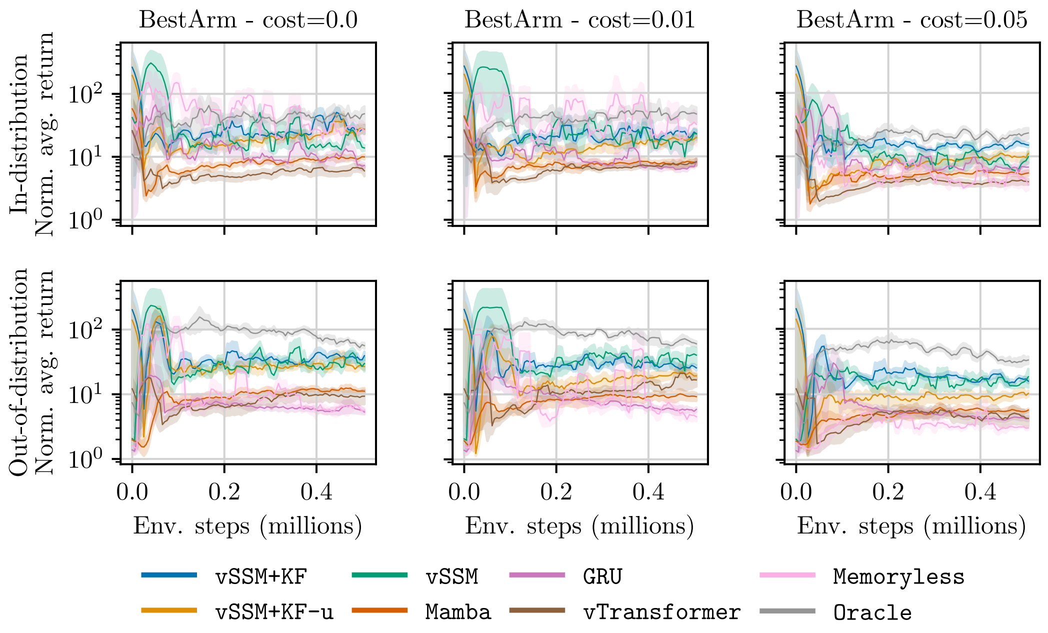


Figure C.2.: Average (log) episode length over 100 episodes in and out of distribution, for increasing costs. We report the mean and standard error over 5 random seeds.

[1]We use a lower value of $-0.35 \log(1/\dim(\mathcal{A}))$ in the `MineSweeper` environment from POPGym, as the default value resulted in divergence during training.

C.3. KF Layer Design Ablation

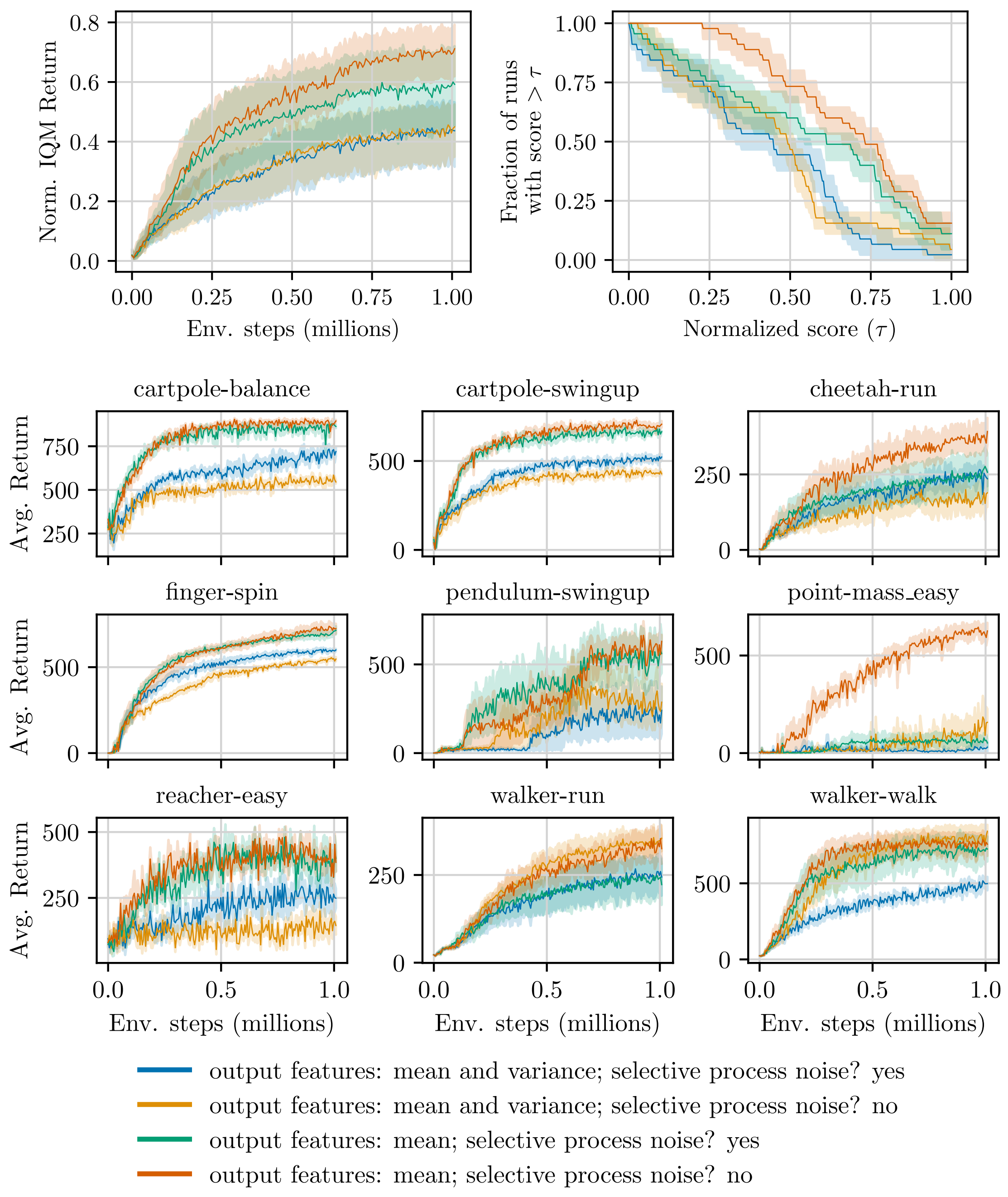


Figure C.3.: Ablation on design considerations for KF layers. **(Top)** Aggregated performance in noisy DMC benchmark (9 tasks) with 95% bootstrap confidence intervals over five random seeds. **(Top-Left)** Inter-quartile mean returns normalized by the score of `Oracle`. **(Top-Right)** Performance profile after 1M environment steps. **(Bottom)** Training curves. We show mean and standard error over five random seeds. Based on these results, our final design for the KF layer uses only the posterior mean state as the output feature and a time-invariant process noise.

C.4. DMC Noise Ablation

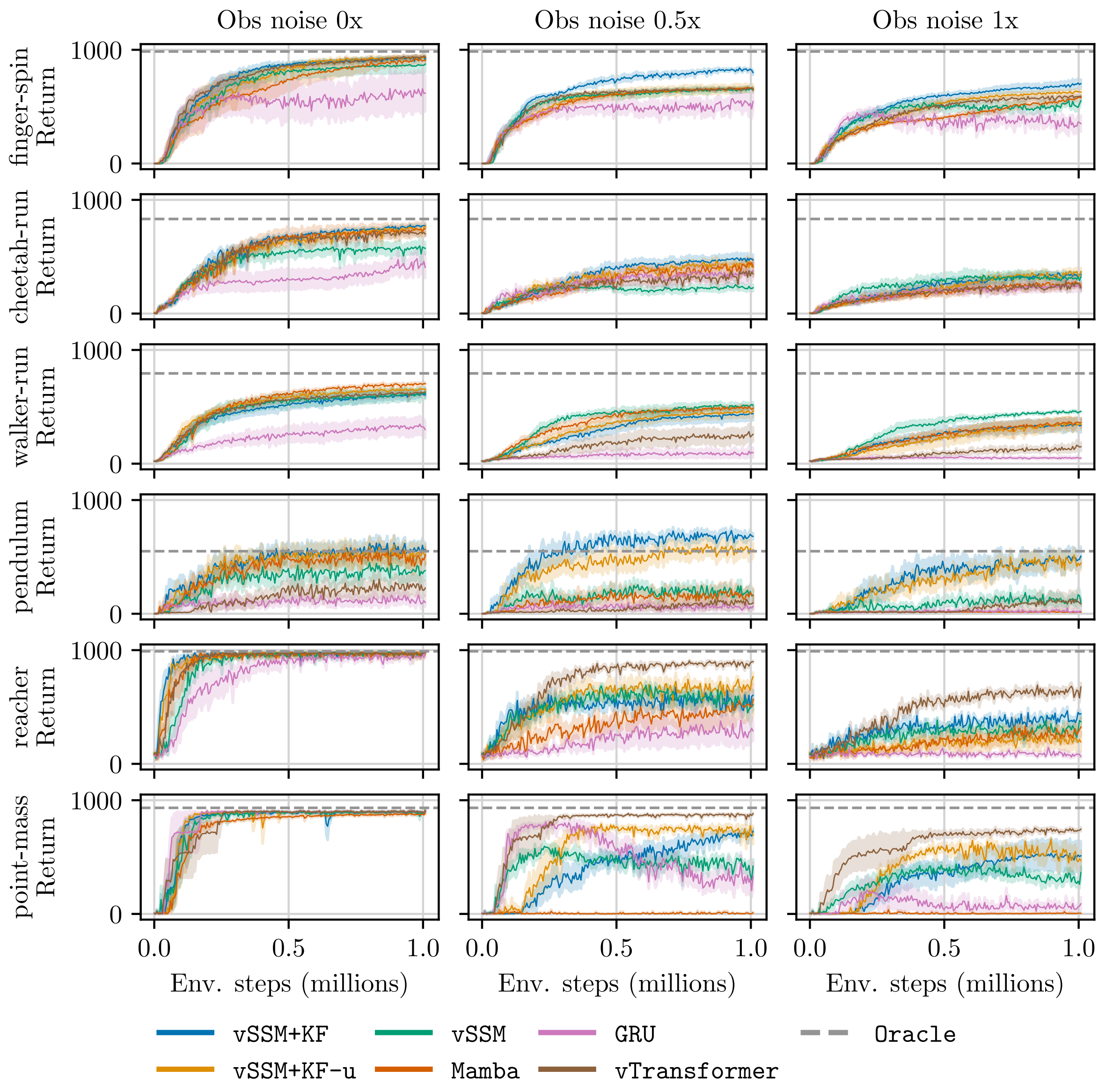


Figure C.4.: Training curves in six environments from the DMC benchmark with increasing levels of noise. We show mean and standard error over five random seeds (ten for `pendulum`). The base noise scale for all tasks is 0.3, except the `pendulum-swingup` and `point-mass` environments where the scale is 0.1

C.5. POPGym Training Curves

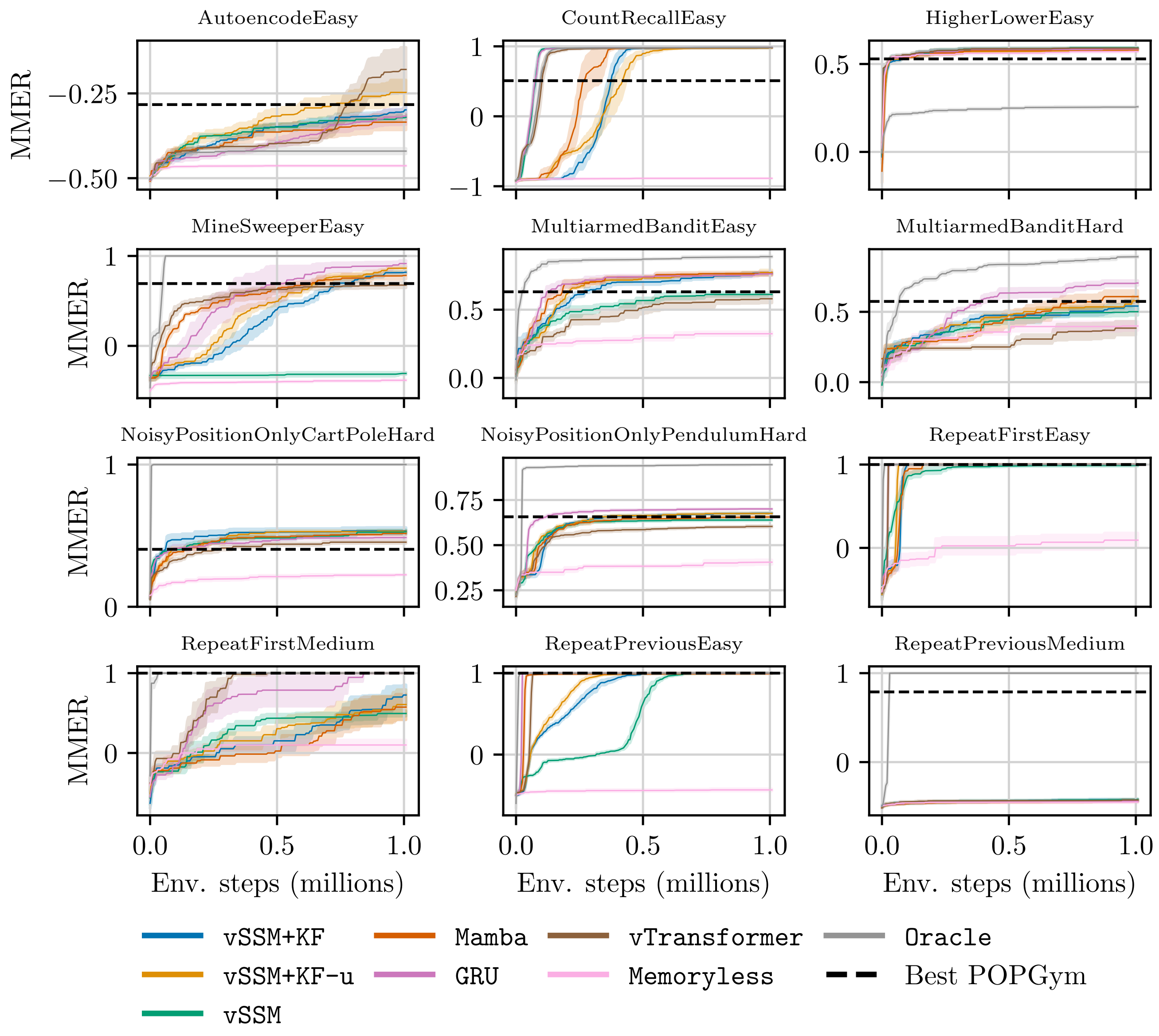


Figure C.5.: POPGym training curves. We show mean and standard error over five random seeds.

C.6. POPGym Scores

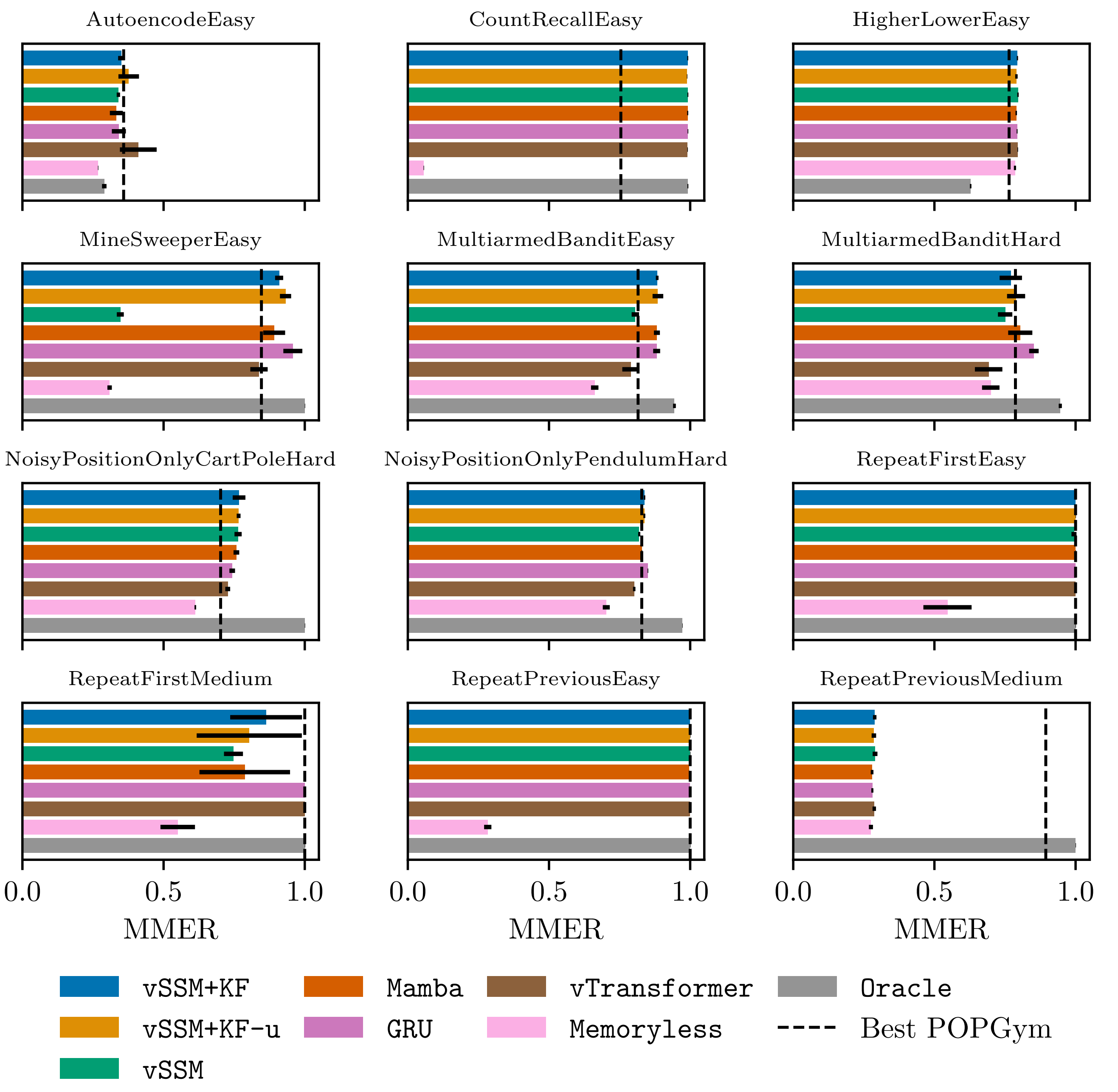


Figure C.6.: POPGym final MMER after 1M training steps. We show mean and standard error over five random seeds.

Table C.2.: Scores on POPGym tasks after 1M environment steps. For each environment, we report the MMER mean and standard error over 5 random seeds after 1M steps of training. The MMER is calculated from 16 test episodes. For reference, we include the best MMER score reported by Morad et al. (2023) (mean and standard deviation over three random seeds). We **bold** the highest score(s) per environment obtained by a sequence model.

	AutoencodeEasy	CountRecallEasy	HigherLowerEasy	MineSweeperEasy
vSSM+KF	-0.299 ± 0.012	$\mathbf{0.983} \pm 0.002$	0.588 ± 0.002	0.818 ± 0.014
vSSM+KF-u	-0.247 ± 0.036	$\mathbf{0.978} \pm 0.001$	0.581 ± 0.004	0.864 ± 0.020
vSSM	-0.320 ± 0.006	$\mathbf{0.984} \pm 0.002$	$\mathbf{0.592} \pm 0.003$	-0.307 ± 0.012
Mamba	-0.335 ± 0.023	0.982 ± 0.002	0.580 ± 0.003	0.783 ± 0.039
GRU	-0.317 ± 0.025	$\mathbf{0.984} \pm 0.001$	0.586 ± 0.002	$\mathbf{0.916} \pm 0.034$
vTransformer	$-\mathbf{0.179} \pm 0.065$	0.982 ± 0.001	$\mathbf{0.590} \pm 0.001$	0.676 ± 0.031
Oracle	-0.420 ± 0.008	0.984 ± 0.002	0.257 ± 0.003	1.000 ± 0.000
Memoryless	-0.463 ± 0.001	-0.887 ± 0.001	0.571 ± 0.004	-0.382 ± 0.008
Best POPGym	-0.283 ± 0.029	0.509 ± 0.062	0.529 ± 0.002	0.693 ± 0.009

	BanditEasy	BanditHard	NoisyCartPoleHard	NoisyPendulumHard
vSSM+KF	$\mathbf{0.766} \pm 0.005$	0.541 ± 0.040	$\mathbf{0.535} \pm 0.023$	0.677 ± 0.003
vSSM+KF-u	$\mathbf{0.771} \pm 0.019$	0.579 ± 0.032	$\mathbf{0.531} \pm 0.007$	0.675 ± 0.003
vSSM	0.612 ± 0.013	0.501 ± 0.025	0.528 ± 0.013	0.639 ± 0.004
Mamba	$\mathbf{0.764} \pm 0.011$	0.608 ± 0.043	$\mathbf{0.516} \pm 0.010$	0.658 ± 0.009
GRU	$\mathbf{0.763} \pm 0.012$	$\mathbf{0.705} \pm 0.017$	0.486 ± 0.010	$\mathbf{0.701} \pm 0.001$
vTransformer	0.580 ± 0.030	0.384 ± 0.049	0.454 ± 0.009	0.604 ± 0.004
Oracle	0.889 ± 0.005	0.892 ± 0.006	1.000 ± 0.000	0.946 ± 0.001
Memoryless	0.324 ± 0.013	0.399 ± 0.031	0.225 ± 0.003	0.406 ± 0.012
Best POPGym	0.631 ± 0.014	0.574 ± 0.049	0.404 ± 0.005	0.657 ± 0.002

	RepeatFirstEasy	RepeatFirstMedium	RepeatPreviousEasy	RepeatPreviousMedium
vSSM+KF	$\mathbf{1.000} \pm 0.000$	0.726 ± 0.127	$\mathbf{1.000} \pm 0.000$	$-\mathbf{0.423} \pm 0.006$
vSSM+KF-u	$\mathbf{1.000} \pm 0.000$	0.607 ± 0.186	$\mathbf{1.000} \pm 0.000$	$-\mathbf{0.429} \pm 0.008$
vSSM	0.989 ± 0.009	0.495 ± 0.034	$\mathbf{1.000} \pm 0.000$	$-\mathbf{0.420} \pm 0.008$
Mamba	$\mathbf{1.000} \pm 0.000$	0.575 ± 0.160	0.993 ± 0.001	-0.441 ± 0.005
GRU	$\mathbf{1.000} \pm 0.000$	$\mathbf{1.000} \pm 0.000$	$\mathbf{1.000} \pm 0.000$	-0.440 ± 0.004
vTransformer	$\mathbf{1.000} \pm 0.000$	$\mathbf{1.000} \pm 0.000$	$\mathbf{1.000} \pm 0.000$	$-\mathbf{0.426} \pm 0.006$
Oracle	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000
Memoryless	0.093 ± 0.085	0.100 ± 0.061	-0.434 ± 0.013	-0.450 ± 0.007
Best POPGym	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.789 ± 0.288

List of Acronyms

CBOP Bayesian Model-Based Value Expansion for Offline Policy Optimization.
CDF Cumulative Distribution Function.
CVaR Conditional Value-At-Risk.
DMC DeepMind Control.
DQN Deep Q-Network.
EQR Epistemic Quantile Regression.
EQR-SAC Epistemic Quantile Regression Soft Actor-Critic.
GP Gaussian Process.
GPU Graphics Processing Unit.
GRU Gated Recurrent Unit.
HiPPO Higher-order Polynomial Projection Operators.
IQM Inter-Quartile Mean.
KF Kalman Filter.
LSTM Long Short-Term Memory.
MAO Masked Associative Operator.
MBPO Model-Based Policy Optimization.
MBRL Model-Based Reinforcement Learning.
MDP Markov Decision Process.
MLP Multi-Layer Perceptron.
MMDP Multi-model Markov Decision Process.
MMER Maximum Mean Episodic Return.
MOPO Model-Based Offline Policy Optimization.
MRP Markov Reward Process.
NN Neural Network.
OFU Optimism in the Face of Uncertainty.
OOD Out of Distribution.
PDF Probability Density Function.
PETS Probabilistic Ensemble Trajectory Sampling.
POMDP Partially Observable Markov Decision Process.
POPGym Partially Observable Process Gym.
PPO Proximal Policy Optimization.
PSRL Probabilistic Sampling Reinforcement Learning.
QR-MBPO Quantile Regression Model-Based Policy Optimization.
QU-SAC Q-Uncertainty Soft Actor-Critic.
RAC Recurrent Actor-Critic.
RKN Recurrent Kalman Network.
RL Reinforcement Learning.
RNN Recurrent Neural Network.

RSSM	Recurrent State-Space Model.
SAC	Soft Actor-Critic.
SSM	State-Space Model.
TD	Temporal Difference.
UBE	Uncertainty Bellman Equation.
UCB	Upper-Confidence Bound.

List of Figures

List of Figures

List of Tables

List of Tables

List of Algorithms

Curriculum Vitae

Education

Ph.D. Candidate — **TU Darmstadt / Bosch Center for AI**
Uncertainty Representations in Reinforcement Learning — 04/2021 - Now

Master of Science — **University of Toronto - Institute for Aerospace Studies**
Multi-agent trajectory planning — 09/2017 - 09/2019

International Exchange Student — **Ecole Polytechnique de Montreal**
Control / Robotics — 09/2015 - 04/2016

Bachelor of Science — **Universidad Simon Bolivar**
Electronics Engineering — 09/2011 - 07/2017

Experience

Research Intern — **Sony AI**
Reinforcement Learning for Game AI — 10/2024 - Now

Software Development Engineer I — **Amazon Prime Air**
Mission-control software for autonomous drone delivery — 10/2019 - 03/2021

Graduate Researcher — **Dynamic Systems Lab**
Multi-agent trajectory planning and UWB localization — 09/2017 - 09/2019

Teaching Assistant — **University of Toronto**
Mathematics for Robotics and Introduction to Robotics — 09/2017 - 12/2018

Research Intern — **Mobile Robotics & Autonomous Systems Lab**
Quadrotor control system — 09/2015 - 08/2016

Publication List

Conference Papers

1. Bottero, A. G., **Luis, C. E.**, Vinogradska, J., Berkenkamp, F., & Peters, J. (2022). “Information-Theoretic Safe Exploration with Gaussian Processes”. Advances in Neural Information Processing Systems (NeurIPS).
2. **Luis, C. E.**, Bottero, A. G., Vinogradska, J., Berkenkamp, F., & Peters, J. (2023). “Model-Based Uncertainty in Value Functions”. International Conference on Artificial Intelligence and Statistics (AISTATS).

Journal Papers

1. Bottero, A. G., **Luis, C. E.**, Vinogradska, J., Berkenkamp, F., & Peters, J. (2023). “Information-Theoretic Safe Bayesian Optimization”. Submitted to Journal of Machine Learning Research (JMLR).
2. **Luis, C. E.**, Bottero, A. G., Vinogradska, J., Berkenkamp, F., & Peters, J. (2023). “Value-Distributional Model-Based Reinforcement Learning”. Journal of Machine Learning Research (JMLR).
3. **Luis, C. E.**, Bottero, A. G., Vinogradska, J., Berkenkamp, F., & Peters, J. (2024). “Model-Based Epistemic Variance of Values for Risk-Aware Policy Optimization”. Submitted to Springer Machine Learning Journal (MLJ).
4. **Luis, C. E.**, Bottero, A. G., Vinogradska, J., Berkenkamp, F., & Peters, J. (2024). “Uncertainty Representations in State-Space Layers for Deep Reinforcement Learning under Partial Observability”. Submitted to Transactions of Machine Learning Research (TMLR).

Workshop Papers

1. **Luis, C. E.**, Bottero, A. G., Vinogradska, J., Berkenkamp, F., & Peters, J. (2023). “Value-Distributional Model-Based Reinforcement Learning”. European Workshop on Reinforcement Learning (EWRL).

Bibliography

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep Reinforcement Learning at the Edge of the Statistical Precipice. In *Advances in Neural Information Processing Systems*, volume 34, pages 29304–29320. Curran Associates, Inc., 2021.

Philip Amortila, Doina Precup, Prakash Panangaden, and Marc G. Bellemare. A Distributional Analysis of Sampling-Based Reinforcement Learning Algorithms. In *International Conference on Artificial Intelligence and Statistics*, volume 108, pages 4357–4366. PMLR, August 2020.

Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-Based Offline Reinforcement Learning with Diversified Q-Ensemble. In *Advances in Neural Information Processing Systems*, volume 34, pages 7436–7447. Curran Associates, Inc., 2021.

Peter Auer and Ronald Ortner. Logarithmic Online Regret Bounds for Undiscounted Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.

Chenjia Bai, Lingxiao Wang, Zhuoran Yang, Zhi-Hong Deng, Animesh Garg, Peng Liu, and Zhaoran Wang. Pessimistic Bootstrapping for Uncertainty-Driven Offline Reinforcement Learning. In *International Conference on Learning Representations*, March 2022.

Bram Bakker. Reinforcement Learning with Long Short-Term Memory. In *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

Philipp Becker and Gerhard Neumann. On Uncertainty in Deep State Space Models for Model-Based Reinforcement Learning. *Transactions on Machine Learning Research*, July 2022.

Philipp Becker, Harit Pandya, Gregor Gebhardt, Cheng Zhao, C. James Taylor, and Gerhard Neumann. Recurrent Kalman Networks: Factorized Inference in High-Dimensional Deep Feature Spaces. In *International Conference on Machine Learning*, volume 97, pages 544–552. PMLR, June 2019.

Philipp Becker, Niklas Freymuth, and Gerhard Neumann. KalMamba: Towards Efficient Probabilistic State Space Models for RL under Uncertainty. In *ICML Workshop on Aligning Reinforcement Learning Experimentalists and Theorists (ARLET)*, June 2024.

Bahram Behzadian, Reazul Hasan Russel, Marek Petrik, and Chin Pang Ho. Optimizing Percentile Criterion using Robust MDPs. In *International Conference on Artificial Intelligence and Statistics*, volume 130, pages 1009–1017. PMLR, April 2021.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47:253–279, June 2013.

Marc G. Bellemare, Will Dabney, and Rémi Munos. A Distributional Perspective on Reinforcement Learning. In *International Conference on Machine Learning*, volume 70, pages 449–458. PMLR, August 2017.

Marc G. Bellemare, Nicolas Le Roux, Pablo Samuel Castro, and Subhodeep Moitra. Distributional Reinforcement Learning with Linear Function Approximation. In *International Conference on Artificial Intelligence and Statistics*, volume 89, pages 2203–2211. PMLR, April 2019.

Marc G Bellemare, Salvatore Candido, Pablo Samuel Castro, Jun Gong, Marlos C Machado, Subhodeep Moitra, Sameera S Ponda, and Ziyu Wang. Autonomous navigation of stratospheric balloons using reinforcement learning. *Nature*, 588(7836):77–82, 2020.

Marc G. Bellemare, Will Dabney, and Mark Rowland. *Distributional Reinforcement Learning*. MIT Press, 2023.

Richard Bellman. *Dynamic Programming*. Princeton University Press, 1957.

Patrick Billingsley. *Probability and Measure*. John Wiley & Sons, third edition edition, 1995.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*, volume 29. Springer, 2006.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym, 2016.

Noam Brown and Tuomas Sandholm. Superhuman AI for Multiplayer Poker. *Science*, 365(6456):885–890, 2019.

Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-Efficient Reinforcement Learning with Stochastic Ensemble Value Expansion. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. TransDreamer: Reinforcement Learning with Transformer World Models. In *DeepRL Workshop NeurIPS*. arXiv, 2021.

Richard Y. Chen, Szymon Sidor, Pieter Abbeel, and John Schulman. UCB Exploration via Q-Ensembles. *arXiv:1706.01502 [cs, stat]*, November 2017.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *8th Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111. Association for Computational Linguistics (ACL), 2014.

Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-Sensitive and Robust Decision-Making: a CVaR Optimization Approach. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

Petros Christodoulou. Soft Actor-Critic for Discrete Action Settings. *arXiv:1910.07207*, October 2019.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Kamil Ciosek, Quan Vuong, Robert Loftin, and Katja Hofmann. Better Exploration with Optimistic Actor Critic. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Sebastian Curi, Felix Berkenkamp, and Andreas Krause. Efficient Model-Based Reinforcement Learning through Optimistic Policy Search and Planning. In *Advances in Neural Information Processing Systems*, volume 33, pages 14156–14170. Curran Associates, Inc., 2020.

Will Dabney, Georg Ostrovski, David Silver, and Remi Munos. Implicit Quantile Networks for Distributional Reinforcement Learning. In *International Conference on Machine Learning*, volume 80, pages 1096–1105. PMLR, July 2018a.

Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional Reinforcement Learning with Quantile Regression. In *AAAI Conference on Artificial Intelligence*, pages 2892–2901. AAAI Press, 2018b.

Richard Dearden, Nir Friedman, and Stuart Russell. Bayesian Q-Learning. In *AAAI Conference on Artificial Intelligence*, pages 761–768, 1998. American Association for Artificial Intelligence.

Richard Dearden, Nir Friedman, and David Andre. Model Based Bayesian Exploration. In *Conference on Uncertainty in Artificial Intelligence*, pages 150–159, 1999. Morgan Kaufmann Publishers Inc.

Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *International Conference on Machine Learning*, pages 465–472, 2011.

Erick Delage and Shie Mannor. Percentile Optimization for Markov Decision Processes with Parameter Uncertainty. *Operations Research*, 58(1):203–213, February 2010.

Stefan Depeweg. *Modeling Epistemic and Aleatoric Uncertainty with Bayesian Neural Networks and Latent Variables*. Dissertation, Technische Universität München, München, 2019.

Stefan Depeweg, Jose-Miguel Hernandez-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning. In *International Conference on Machine Learning*, pages 1184–1193. PMLR, July 2018.

Esther Derman, Daniel J Mankowitz, Timothy A Mann, and Shie Mannor. Soft-Robust Actor-Critic Policy-Gradient. In *Uncertainty in Artificial Intelligence Conference*, pages 208–218, 2018.

Esther Derman, Daniel Mankowitz, Timothy Mann, and Shie Mannor. A Bayesian Approach to Robust Reinforcement Learning. In *Uncertainty in Artificial Intelligence Conference*, volume 115, pages 648–658. PMLR, July 2020.

Omar Darwiche Domingues, Yannis Flet-Berliac, Edouard Leurent, Pierre Ménard, Xuedong Shang, and Michal Valko. rlberry - A Reinforcement Learning Library for Research and Education, October 2021.

Yaakov Engel, Shie Mannor, and Ron Meir. Bayes Meets Bellman: The Gaussian Process Approach to Temporal Difference Learning. In *International Conference on Machine Learning*, pages 154–161. AAAI Press, 2003.

Hannes Eriksson, Debabrota Basu, Mina Alibeigi, and Christos Dimitrakakis. SENTINEL: taming uncertainty with ensemble based distributional reinforcement learning. In *Conference on Uncertainty in Artificial Intelligence*, volume 180, pages 631–640. PMLR, August 2022.

Benjamin Eysenbach, Alexander Khazatsky, Sergey Levine, and Ruslan Salakhutdinov. Mismatched no more: joint model-policy optimization for model-based RL. In *Neural Information Processing Systems*, 2022. Curran Associates Inc.

Ying Fan and Yifei Ming. Model-based Reinforcement Learning for Continuous Control with Posterior Sampling. In *International Conference on Machine Learning*, volume 139, pages 3078–3087. PMLR, July 2021.

Mattie Fellows, Kristian Hartikainen, and Shimon Whiteson. Bayesian Bellman Operators. In *Advances in Neural Information Processing Systems*, volume 34, pages 13641–13656. Curran Associates, Inc., 2021.

Meire Fortunato, Melissa Tan, Ryan Faulkner, Steven Hansen, Adrià Puigdomènech Badia, Gavin Buttimore, Charles Deck, Joel Z Leibo, and Charles Blundell. Generalization of Reinforcement Learners with Working and Episodic Memory. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Lukas Froehlich, Maksym Lefarov, Melanie Zeilinger, and Felix Berkenkamp. On-Policy Model Errors in Reinforcement Learning. In *International Conference on Learning Representations*, 2022.

Daniel Y. Fu, Tri Dao, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Ré. Hungry Hungry Hippos: Towards Language Modeling with State Space Models. In *International Conference on Learning Representations*, 2023.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for Deep Data-Driven Reinforcement Learning, 2020.

Scott Fujimoto, Herke van Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *International Conference on Machine Learning*, volume 80, pages 1587–1596. PMLR, July 2018.

Yarin Gal. *Uncertainty in deep learning*. Dissertation, University of Cambridge, 2016.

Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian Reinforcement Learning: A Survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015.

Karan Goel, Albert Gu, Chris Donahue, and Christopher Re. It's Raw! Audio Generation with State-Space Models. In *International Conference on Machine Learning*, volume 162, pages 7616–7633. PMLR, July 2022.

Albert Gu and Tri Dao. Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *arXiv:2312.00752*, 2023.

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. HiPPO: Recurrent Memory with Optimal Polynomial Projections. In *Advances in Neural Information Processing Systems*, volume 33, pages 1474–1487. Curran Associates, Inc., 2020.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently Modeling Long Sequences with Structured State Spaces. In *International Conference on Learning Representations*, 2022a.

Albert Gu, Ankit Gupta, Karan Goel, and Christopher Ré. On the Parameterization and Initialization of Diagonal State Space Models. In *Advances in Neural Information Processing Systems*, volume 35, 2022b.

David Ha and Jürgen Schmidhuber. Recurrent World Models Facilitate Policy Evolution. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Tuomas Haarnoja, Anurag Ajay, Sergey Levine, and Pieter Abbeel. Backprop KF: Learning Discriminative Deterministic State Estimators. In *Advances in Neural Information Processing Systems*, pages 4383–4391, 2016. Curran Associates Inc.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning*, volume 80, pages 1861–1870. PMLR, July 2018.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic Algorithms and Applications. *arXiv:1812.05905*, January 2019.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning Latent Dynamics for Planning from Pixels. In *International Conference on Machine Learning*, volume 97, pages 2555–2565. PMLR, June 2019.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to Control: Learning Behaviors by Latent Imagination. In *International Conference on Learning Representations*, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering Diverse Domains through World Models, January 2023.

Matthew Hausknecht and Peter Stone. Deep Recurrent Q-Learning for Partially Observable MDPs. In *AAAI Fall Symposium Series*, 2015.

Nicolas Heess, Jonathan J. Hunt, Timothy P. Lillicrap, and David Silver. Memory-Based Control with Recurrent Neural Networks. In *NIPS Deep Reinforcement Learning Workshop*, December 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal Regret Bounds for Reinforcement Learning. *Journal of Machine Learning Research*, 11(4), 2010.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to Trust Your Model: Model-Based Policy Optimization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Jihwan Jeong, Xiaoyu Wang, Michael Gimelfarb, Hyunwoo Kim, Baher Abdulhai, and Scott Sanner. Conservative Bayesian Model-Based Value Expansion for Offline Policy Optimization. In *International Conference on Learning Representations*, February 2023.

Emilio Jorge, Hannes Eriksson, Christos Dimitrakakis, Debabrota Basu, and Divya Grover. Inferential Induction: A Novel Framework for Bayesian Reinforcement Learning. In *Proceedings on "I Can't Believe It's Not Better!" at NeurIPS Workshops*, volume 137, pages 43–52. PMLR, December 2020.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101(1):99–134, 1998.

R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82 (1):35–45, March 1960.

Gabriel Kalweit and Joschka Boedecker. Uncertainty-driven Imagination for Continuous Deep Reinforcement Learning. In *Conference on Robot Learning*, pages 195–206. PMLR, October 2017.

Tyler Kastner, Murat A Erdogdu, and Amir-massoud Farahmand. Distributional Model Equivalence for Risk-Sensitive Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 56531–56552. Curran Associates, Inc., 2023.

Ramtin Keramati, Christoph Dann, Alex Tamkin, and Emma Brunskill. Being Optimistic to Be Conservative: Quickly Learning a CVaR Policy. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 4436–4443, April 2020.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOReL: Model-Based Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 21810–21823. Curran Associates, Inc., 2020.

Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? Does it matter? *Structural Safety*, 31(2): 105–112, March 2009.

Ilya Kostrikov. PyTorch Implementations of Reinforcement Learning Algorithms, 2018.

Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative Q-Learning for Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 1179–1191. Curran Associates, Inc., 2020.

Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics. In *International Conference on Machine Learning*, volume 119, pages 5556–5566. PMLR, July 2020.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective Mismatch in Model-based Reinforcement Learning. In *Learning for Dynamics and Control*, pages 761–770. PMLR, 2020.

Nathan Lambert, Kristofer Pister, and Roberto Calandra. Investigating Compounding Prediction Errors in Learned Dynamics Models. *arXiv:2203.09637 [cs]*, March 2022.

Kimin Lee, Michael Laskin, Aravind Srinivas, and Pieter Abbeel. SUNRISE: A Simple Unified Framework for Ensemble Learning in Deep Reinforcement Learning. In *Proceedings of the 38th International Conference on Machine Learning*, pages 6131–6141. PMLR, July 2021.

Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-Online Reinforcement Learning via Balanced Replay and Pessimistic Q-Ensemble. In *Conference on Robot Learning*, pages 1702–1712. PMLR, January 2022.

Kun Lei, Zhengmao He, Chenhao Lu, Kaizhe Hu, Yang Gao, and Huazhe Xu. Uni-O4: Unifying Online and Offline Deep Reinforcement Learning with Multi-Step On-Policy Optimization. In *International Conference on Learning Representations*, 2024.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv:2005.01643 [cs, stat]*, November 2020.

Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A Contextual-Bandit Approach to Personalized News Article Recommendation. In *International Conference on World Wide Web*, pages 661–670, 2010. Association for Computing Machinery.

Long-Ji Lin and Tom M Mitchell. Reinforcement Learning with Hidden States. In *International Conference on Simulation of Adaptive Behavior*, pages 271–280. MIT Press, 1993.

Qinghua Liu, Alan Chung, Csaba Szepesvari, and Chi Jin. When Is Partially Observable Reinforcement Learning Not Scary? In *Conference on Learning Theory*, pages 5175–5220. PMLR, June 2022.

Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. Structured State Space Models for In-Context Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 47016–47031. Curran Associates, Inc., 2023.

Carlos E. Luis, Alessandro G. Bottero, Julia Vinogradska, Felix Berkenkamp, and Jan Peters. Model-Based Uncertainty in Value Functions. In *International Conference on Artificial Intelligence and Statistics*, volume 206, pages 8029–8052. PMLR, April 2023a.

Carlos E. Luis, Alessandro G. Bottero, Julia Vinogradska, Felix Berkenkamp, and Jan Peters. Value-Distributional Model-Based Reinforcement Learning. *arXiv:2308.06590*, August 2023b.

Carlos E. Luis, Alessandro G. Bottero, Julia Vinogradska, Felix Berkenkamp, and Jan Peters. Model-Based Epistemic Variance of Values for Risk-Aware Policy Optimization. *arXiv:2312.04386*, September 2024a.

Carlos E. Luis, Alessandro G. Bottero, Julia Vinogradska, Felix Berkenkamp, and Jan Peters. Uncertainty Representations in State-Space Layers for Deep Reinforcement Learning under Partial Observability. *arXiv:2409.16824*, September 2024b.

Clare Lyle, Marc G. Bellemare, and Pablo Samuel Castro. A Comparative Analysis of Expected and Distributional Reinforcement Learning. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 4504–4511, July 2019.

Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, and Luke Zettlemoyer. Mega: Moving Average Equipped Gated Attention. In *International Conference on Learning Representations*, 2023.

Ali Malik, Volodymyr Kuleshov, Jiaming Song, Danny Nemer, Harlan Seymour, and Stefano Ermon. Calibrated Model-Based Deep Reinforcement Learning. In *International Conference on Machine Learning*, pages 4314–4323. PMLR, May 2019.

Efstratios Markou and Carl E Rasmussen. Bayesian Methods for Efficient Reinforcement Learning in Tabular Problems. In *NeurIPS Workshop on Biological and Artificial RL*, 2019.

Alberto Maria Metelli, Amarildo Likmeta, and Marcello Restelli. Propagating Uncertainty in Reinforcement Learning via Wasserstein Barycenters. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. In *NIPS Deep Learning Workshop*, December 2013.

Steven Morad, Ryan Kortvelesy, Matteo Bettini, Stephan Liwicki, and Amanda Prorok. POPGym: Benchmarking Partially Observable Reinforcement Learning. In *International Conference on Learning Representations*, 2023.

Ted Moskovitz, Jack Parker-Holder, Aldo Pacchiano, Michael Arbel, and Michael Jordan. Tactical Optimism and Pessimism for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 12849–12863. Curran Associates, Inc., 2021.

Kevin Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

Mitsuhiko Nakamoto, Yuexiang Zhai, Anikait Singh, Max Sobol Mark, Yi Ma, Chelsea Finn, Aviral Kumar, and Sergey Levine. Cal-QL: Calibrated Offline RL Pre-Training for Efficient Online Fine-Tuning. In *Neural Information Processing Systems*, November 2023.

Eric Nguyen, Karan Goel, Albert Gu, Gordon Downs, Preey Shah, Tri Dao, Stephen Baccus, and Christopher Ré. S4ND: Modeling Images and Videos as Multidimensional Signals with State Spaces. In *Advances in Neural Information Processing Systems*, volume 35, pages 2846–2861. Curran Associates, Inc., 2022.

Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent Model-Free RL Can Be a Strong Baseline for Many POMDPs. In *International Conference on Machine Learning*, pages 16691–16723. PMLR, June 2022.

Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. When Do Transformers Shine in RL? Decoupling Memory from Credit Assignment. In *Advances in Neural Information Processing Systems*, volume 36, pages 50429–50452. Curran Associates, Inc., 2023.

Tianwei Ni, Benjamin Eysenbach, Erfan Seyedsalehi, Michel Ma, Clement Gehring, Aditya Mahajan, and Pierre-Luc Bacon. Bridging State and History Representations: Understanding Self-Predictive RL. In *International Conference on Learning Representations*, 2024.

Brendan O'Donoghue. Variational Bayesian Reinforcement Learning with Regret Bounds. In *Advances in Neural Information Processing Systems*, volume 34, pages 28208–28221. Curran Associates, Inc., 2021.

Brendan O'Donoghue, Ian Osband, Remi Munos, and Volodymyr Mnih. The Uncertainty Bellman Equation and Exploration. In *International Conference on Machine Learning*, pages 3836–3845, 2018.

Brendan O'Donoghue, Ian Osband, and Catalin Ionescu. Making Sense of Reinforcement Learning and Probabilistic Inference. In *International Conference on Learning Representations*, September 2019.

Junhyuk Oh, Valliappa Chockalingam, Satinder, and Honglak Lee. Control of Memory, Active Perception, and Action in Minecraft. In *International Conference on Machine Learning*, volume 48, pages 2790–2799, June 2016. PMLR.

Ian Osband and Benjamin Van Roy. Why is Posterior Sampling Better than Optimism for Reinforcement Learning? In *International Conference on Machine Learning*, pages 2701–2710. PMLR, 2017.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (More) Efficient Reinforcement Learning via Posterior Sampling. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep Exploration via Bootstrapped DQN. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized Prior Functions for Deep Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Ian Osband, Benjamin Van Roy, Daniel J Russo, and Zheng Wen. Deep Exploration via Randomized Value Functions. *Journal of Machine Learning Research*, 20:1–62, 2019.

Christos H. Papadimitriou and John N. Tsitsiklis. The Complexity of Markov Decision Processes. *Mathematics of Operations Research*, 12(3):441–450, 1987.

Emilio Parisotto and Russ Salakhutdinov. Efficient Transformers in Reinforcement Learning using Actor-Learner Distillation. In *International Conference on Learning Representations*, October 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Luis Pineda, Brandon Amos, Amy Zhang, Nathan O. Lambert, and Roberto Calandra. MBRL-Lib: A Modular Library for Model-based Reinforcement Learning. *arXiv:2104.10159 [cs, eess]*, April 2021.

Antonin Raffin, Jens Kober, and Freek Stulp. Smooth Exploration for Robotic Reinforcement Learning, June 2021.

Marc Rigter, Bruno Lacerda, and Nick Hawes. RAMBO-RL: Robust Adversarial Model-Based Offline Reinforcement Learning. In *Advances in Neural Information Processing Systems*, October 2022.

Mark Rowland, Marc Bellemare, Will Dabney, Remi Munos, and Yee Whye Teh. An Analysis of Categorical Distributional Reinforcement Learning. In *International Conference on Artificial Intelligence and Statistics*, volume 84, pages 29–37. PMLR, April 2018.

Mark Rowland, Robert Dadashi, Saurabh Kumar, Remi Munos, Marc G. Bellemare, and Will Dabney. Statistics and Samples in Distributional Reinforcement Learning. In *International Conference on Machine Learning*, volume 97, pages 5528–5536. PMLR, June 2019.

Mark Rowland, Rémi Munos, Mohammad Gheshlaghi Azar, Yunhao Tang, Georg Ostrovski, Anna Harutyunyan, Karl Tuyls, Marc G. Bellemare, and Will Dabney. An Analysis of Quantile Temporal-Difference Learning, January 2023.

Mohammad Reza Samsami, Artem Zholus, Janarthanan Rajendran, and Sarath Chandar. Mastering Memory Tasks with World Models. In *International Conference on Learning Representations*, 2024.

Simo Sarkka and Angel F. Garcia-Fernandez. Temporal Parallelization of Bayesian Smoothers. *IEEE Transactions on Automatic Control*, 66(1):299–306, January 2021.

J. Schmidhuber. Curious Model-Building Control Systems. In *IEEE International Joint Conference on Neural Networks*, pages 1458–1463 vol.2, 1991. IEEE.

Jurgen Schmidhuber. Networks Adjusting Networks. In *Distributed Adaptive Neural Information Processing*, pages 197–208, 1990.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv:1707.06347*, August 2017.

Vaisakh Shaj, Philipp Becker, Dieter Büchler, Harit Pandya, Niels van Duijkeren, C. James Taylor, Marc Hanheide, and Gerhard Neumann. Action-Conditional Recurrent Kalman Networks For Forward and Inverse Dynamics Learning. In *Conference on Robot Learning*, volume 155, pages 765–781. PMLR, November 2021a.

Vaisakh Shaj, Dieter Büchler, Rohit Sonker, Philipp Becker, and Gerhard Neumann. Hidden Parameter Recurrent State Space Models For Changing Dynamics Scenarios. In *International Conference on Learning Representations*, October 2021b.

Vaisakh Shaj, Saleh Gholam Zadeh, Ozan Demir, Luiz Ricardo Douat, and Gerhard Neumann. Multi Time Scale World Models. In *Advances in Neural Information Processing Systems*, November 2023.

Jimmy T. H. Smith, Andrew Warrington, and Scott Linderman. Simplified State Space Layers for Sequence Modeling. In *International Conference on Learning Representations*, 2023.

Matthew J Sobel. The Variance of Discounted Markov Decision Processes. *Journal of Applied Probability*, 19 (4):794–802, 1982.

Lauren N. Steimle, David L. Kaufman, and Brian T. Denton. Multi-model Markov decision processes. *IISE Transactions*, pages 1–16, May 2021.

Alexander L Strehl and Michael L Littman. An Analysis of Model-Based Interval Estimation for Markov Decision Processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.

Xihong Su and Marek Petrik. Solving multi-model MDPs by coordinate ascent and dynamic programming. In *Conference on Uncertainty in Artificial Intelligence*, volume 216, pages 2016–2025. PMLR, August 2023.

Richard Sutton and Andrew Barto. *Reinforcement Learning: An Introduction*, volume 7. MIT Press, 2018.

Richard S. Sutton. Dyna, an Integrated Architecture for Learning, Planning, and Reacting. *ACM SIGART Bulletin*, 2(4):160–163, July 1991.

Aviv Tamar, Dotan Di Castro, and Shie Mannor. Temporal Difference Methods for the Variance of the Reward To Go. In *International Conference on Machine Learning*, pages 495–503. PMLR, 2013.

Gerald Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38(3):58–68, 1995.

Daniil Tiapkin, Denis Belomestny, Eric Moulines, Alexey Naumov, Sergey Samsonov, Yunhao Tang, Michal Valko, and Pierre Menard. From Dirichlet to Rubin: Optimistic Exploration in RL without Bonuses. In *Proceedings of the 39th International Conference on Machine Learning*, pages 21380–21431. PMLR, June 2022.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IEEE/RSJ International Conference on Intelligent Robots and systems*, pages 5026–5033. IEEE, 2012.

Saran Tunyasuvunakool, Alistair Muldal, Yotam Doron, Siqi Liu, Steven Bohez, Josh Merel, Tom Erez, Timothy Lillicrap, Nicolas Heess, and Yuval Tassa. dm_control: Software and Tasks for Continuous Control. *Software Impacts*, 6:100022, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Cédric Villani. *Optimal Transport: Old and New*, volume 338. Springer, 2008.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, and others. Grandmaster Level in StarCraft II using Multi-Agent Reinforcement Learning. *Nature*, 575(7782):350–354, 2019.

Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Science & Business Media, 2013.

Daan Wierstra, Alexander Foerster, Jan Peters, and Jürgen Schmidhuber. Solving Deep Memory POMDPs with Recurrent Policy Gradients. In *Artificial Neural Networks*, pages 697–706, 2007. Springer Berlin Heidelberg.

Peter R. Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J. Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, Leilani Gilpin, Piyush Khandelwal, Varun Kompella, HaoChih Lin, Patrick MacAlpine, Declan Oller, Takuma Seno, Craig Sherstan, Michael D. Thomure, Houmehr Aghabozorgi, Leon Barrett, Rory Douglas, Dion Whitehead, Peter Dürr, Peter Stone, Michael Spranger, and Hiroaki Kitano. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature*, 602(7896):223–228, February 2022.

Derek Yang, Li Zhao, Zichuan Lin, Tao Qin, Jiang Bian, and Tie-Yan Liu. Fully Parameterized Quantile Function for Distributional Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based Offline Policy Optimization. In *Advances in Neural Information Processing Systems*, volume 33, pages 14129–14142. Curran Associates, Inc., 2020.

Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. COMBO: Conservative Offline Model-Based Policy Optimization. In *Advances in Neural Information Processing Systems*, volume 34, pages 28954–28967. Curran Associates, Inc., 2021.

Biao Zhang and Rico Sennrich. Root Mean Square Layer Normalization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Kai Zhao, Yi Ma, Jianye Hao, Jinyi Liu, Yan Zheng, and Zhaopeng Meng. Improving Offline-to-Online Reinforcement Learning with Q-Ensembles. In *Workshop on New Frontiers in Learning, Control, and Dynamical Systems at the International Conference on Machine Learning*. arXiv, December 2023.

Bo Zhou, Hongsheng Zeng, Fan Wang, Yunxiang Li, and Hao Tian. Efficient and Robust Reinforcement Learning with Uncertainty-based Value Expansion. *arXiv:1912.05328 [cs]*, December 2019.

Qi Zhou, HouQiang Li, and Jie Wang. Deep Model-Based Reinforcement Learning via Estimated Uncertainty and Conservative Policy Optimization. In *AAAI Conference on Artificial Intelligence*, volume 34, pages 6941–6948, April 2020.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. In *International Conference on Robotics and Automation*, 2017.

Luisa Zintgraf, Sebastian Schulze, Cong Lu, Leo Feng, Maximilian Igl, Kyriacos Shiarlis, Yarin Gal, Katja Hofmann, and Shimon Whiteson. VariBAD: Variational Bayes-Adaptive Deep RL via Meta-Learning. *Journal of Machine Learning Research*, 22(289):1–39, 2021.