TU Delft

Delft University of Technology

Document Version
Final published version

Citation (APA)
van der Vaart, P. R. (2026). *Bayesian Model-Free Deep Reinforcement Learning*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:74de7a79-a77e-4e18-a36e-babe330131bc

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

Bayesian Model-Free Deep Reinforcement Learning

Pascal R. van der Vaart

BAYESIAN MODEL-FREE DEEP REINFORCEMENT LEARNING

BAYESIAN MODEL-FREE DEEP REINFORCEMENT LEARNING

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology
by the authority of the Rector Magnificus, Prof. dr. ir. H. Bijl,
chair of the Board for Doctorates
to be defended publicly on Friday 22 May 2026 at 10:00

by

Pascal Robert VAN DER VAART

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus	chairperson
Prof. dr. M. T. J. Spaan	Delft University of Technology, *promotor*
Dr. N. Yorke-Smith	Delft University of Technology, *promotor*
Independent members:	
Prof. dr. F. A. Oliehoek	Delft University of Technology
Dr. ir. G. N. J. C. Bierkens	Delft University of Technology
Prof. dr. N. Jansen	Ruhr University Bochum, Germany
Prof. dr. V. Fortuin	TU Nuremberg, Germany
Prof. dr. M. M. de Weerdt	Delft University of Technology, *reserve member*

Keywords: Bayesian, reinforcement learning, machine learning

Cover by: Pascal Robert van der Vaart

Copyright © 2026 by Pascal Robert Van der Vaart

ISBN 978-94-6384-961-6

An electronic copy of this dissertation is available at
`https://repository.tudelft.nl/`.

for my family, in the broadest sense of the word

CONTENTS

SUMMARY

The goal of reinforcement learning is to train agents to perform tasks under little supervision. Tasks are specified by a reward function and transition function, which state how much reward the agent gets for its action in a state, and how the environment state changes based on the action the agent took. Typically reinforcement learning assumes no prior knowledge over the reward and transition function, meaning that agents need to explore the environment and learn essentially through trial and error. Model-free methods attempt to learn which actions lead to good outcomes without modeling the reward or environments itself. Efficiently selecting that actions are promising is an active research direction which can greatly reduce the number of total interactions needed for an agent to learn the task, potentially opening the door to new applications where trials or simulations are expensive or compute is limited.

Uncertainty quantification is a central mechanism in such efficient exploration methods. Provided with an estimate of how certain the agent is about the outcome of an action, it can intelligently weigh whether it is worth exploring. The Bayesian paradigm is one method to quantify uncertainty in machine learning. It models the uncertainty with a probability distributions over models, specifying how likely a model is based on the data the agent has collected.

We adopt a Bayesian point of view in model-free reinforcement learning, and develop a deeper understanding on when Bayesian reinforcement learning methods can be expected to work well and challenges that remain. To this end, in Chapter 2 we propose training ensembles through Sequential Monte Carlo, obtaining a sample from the posterior distribution of a deep Q-learning agent. We observe that agents are able to perform directed exploration, although not necessarily more efficiently than standard ensembles in every environment. Furthermore, in Chapter 3 we theoretically analyze existing Bayesian Deep model-Free Reinforcement Learning methods, and unify them into a single theoretical framework we call Epistemic Bellman Operators. We prove that these operators are contractions, establishing convergence of derived algorithms in a simplified setting. Finally, in Chapter 4 we analyze the likelihood and prior assumptions in existing Bayesian deep model-free reinforcement learning methods, and find through statistical tests that the standard likelihood assumptions are violated on every benchmark we tested. We also find that we can improve performance of Bayesian model-free reinforcement learning methods by picking different priors based on empirical data from unrelated tasks, which transfer to new environments.

This dissertation establishes several desirable properties of Bayesian Deep model-free reinforcement learning, but also raises some key issues, most notably misspecification in Chapter 4. We hope our findings convince other Bayesian reinforcement learning researchers to give more attention to assumptions about priors and likelihoods.

SAMENVATTING

Het doel van reinforcement learning is het trainen van agenten om taken uit te voeren met minimale supervisie. Taken worden gespecificeerd door een beloningsfunctie en een transitiefunctie, die aangeven hoeveel beloning de agent ontvangt voor zijn actie in een bepaalde toestand, en hoe de omgevingstoestand verandert op basis van de actie die de agent heeft ondernomen. Reinforcement learning gaat doorgaans uit van geen voorkennis over de belonings- en transitiefunctie, wat betekent dat agenten de omgeving moeten verkennen en in wezen moeten leren door middel van trial and error. Modelvrije methoden proberen te leren welke acties tot goede uitkomsten leiden, zonder de beloning of de omgeving precies te modelleren. Het efficiënt selecteren van veelbelovende acties is een actieve onderzoeksrichting die het totale aantal benodigde interacties voor een agent om een taak te leren aanzienlijk kan verminderen, waardoor mogelijk nieuwe toepassingen mogelijk worden waarbij experimenten of simulaties kostbaar zijn of de rekenkracht beperkt is.

Kwantificering van onzekerheid is een centraal mechanisme in dergelijke efficiënte exploratiemethoden. Met een schatting van hoe zeker de agent is over de uitkomst van een actie, kan hij op intelligente wijze afwegen of het de moeite waard is om te verkennen. Het Bayesiaans paradigma is één methode om onzekerheid in kunstmatige intelligentie te kwantificeren; het modelleert de onzekerheid met een kansverdeling over modellen, waarbij wordt aangegeven hoe waarschijnlijk een model is op basis van de data die de agent heeft verzameld.

Wij nemen een Bayesiaans standpunt in bij modelvrij reinforcement learning en ontwikkelen een dieper inzicht in wanneer Bayesiaanse reinforcement learning-methoden naar verwachting goed werken en welke uitdagingen er nog resteren. In dat kader stellen we in Hoofdstuk 2 voor om ensembles te trainen via Sequential Monte Carlo, waarbij een steekproef wordt verkregen uit de posteriorverdeling van een deep Q-learning agent. We stellen vast dat agenten gerichte exploratie kunnen uitvoeren, hoewel niet noodzakelijkerwijs efficiënter dan standaard ensembles in elke taak. Verder analyseren we in Hoofdstuk 3 theoretisch bestaande Bayesiaanse diepe modelvrije reinforcement learning-methoden en verenigen we deze in één theoretisch raamwerk dat we Epistemic Bellman Operators noemen. We bewijzen dat deze operatoren contracties zijn, waarmee de convergentie van afgeleide algoritmen in een vereenvoudigde omgeving wordt aangetoond. Tot slot analyseren we in Hoofdstuk 4 de aannames in bestaande diepe Bayesiaanse modelvrije reinforcement learning-methoden, en stellen via statistische toetsen vast dat de standaard aannames worden geschonden in elk benchmark dat we hebben getest. We stellen ook vast dat we de prestaties van Bayesiaanse modelvrije reinforcement learning-methoden kunnen verbeteren door andere priors te kiezen op basis van empirische gegevens uit niet-gerelateerde taken.

Dit proefschrift stelt verschillende wenselijke eigenschappen van Bayesiaanse diepe modelvrije reinforcement learning vast, maar brengt ook een aantal belangrijke kwes-

ties aan het licht, met name misspecificatie van aannames in Hoofdstuk 4. We hopen dat onze bevindingen andere Bayesiaanse reinforcement learning-onderzoekers ervan overtuigen zich meer te richten op de onderliggende aannames.

1

INTRODUCTION

1

Over the past decades, machine learning (ML) has experienced rapid and transformative progress, driven by advances in computational hardware, the proliferation of large-scale datasets, and innovations in algorithm design. Deep learning algorithms have achieved human-competitive or even superhuman performance in tasks such as image classification [He et al., 2016, Dosovitskiy et al., 2020], speech recognition [Hannun et al., 2014], and language modeling [Brown et al., 2020]. For example, over the duration of my PhD, large language models have gone from a curiosity to tools like ChatGPT that now see millions of users daily. Alongside these advances, there has been interest in *Bayesian* approaches to machine learning, to provide principled uncertainty quantification alongside the strong predictive power of deep learning [Papamarkou et al., 2024].

At a foundational level, machine learning algorithms can be broadly classified into three categories: supervised learning, unsupervised learning and reinforcement learning. The distinction is mostly based in the data and kind of feedback they receive. In supervised learning, models are trained on labeled datasets, where each example consists of an input-output pair. The goal is to learn a function that maps inputs to outputs with high accuracy, typically by using an optimization algorithm to minimize a loss function. This paradigm has been particularly successful in domains with abundant annotated data, such as classification tasks in computer vision. Unsupervised learning on the other hand learn patterns from unlabeled data potentially allowing to use much larger data sets as data does not have to be labeled by hand. Finally, reinforcement learning typically does not have access to a data set, and instead learns through interactions with the environment.

1.1. REINFORCEMENT LEARNING

Broadly speaking, reinforcement learning problems are defined by a state space, action space, transition function, reward function, and a discount factor [Sutton and Barto, 2018]. The agent chooses an action depending on the state the environment is in, which will yield a reward depending on the state and action, and transitions the environment to a new state. The agent's goal is to learn a mapping from states to actions, typically called a policy, that maximizes the cumulative future reward. This means that agents need to consider the impact of their actions on future rewards rather than just the immediate rewards. Unlike supervised learning, where algorithms have access to a data set, reinforcement learning methods aim to learn a policy purely by trial and error in the environment or simulator.

When reinforcement learning is combined with deep neural networks as models, the field is referred to as *deep reinforcement learning*. Neural networks are excellent at modeling high dimensional problems and have also enabled deep reinforcement learning to scale to high-dimensional state-action spaces [Mnih et al., 2015, Schrittwieser et al., 2020].

This framework is very general and encapsulates many potential applications. A good example of this is a self-driving car. The car is in an environment and has sensors and cameras, and based on what it perceives (the state) it needs to make decisions such as steering or braking (actions) to get to a goal position (the task). The car receives rewards based on the state and taken actions. For example, breaking a traffic rule might incur a large negative reward, while reaching the goal position yields a positive reward.

The car could be given a small negative reward at each timestep to encourage reaching the goal as fast as possible. Rather than training such a system with a large set of (human) data, reinforcement learning algorithms will drive the car autonomously, learning how decisions impact the state of the vehicle and how that relates to getting to the goal.

1.1.1. MODEL-FREE OR MODEL-BASED?

Reinforcement learning in turn can be categorized by "model-based" and "model-free" approaches. Confusingly, both approaches typically construct models, but the differences lie in what is modeled and how the models are used.

Model-based approaches attempt to model the environment, i.e., the transition and reward functions, and typically try to plan ahead by unrolling their model. On the other hand, model-free algorithms attempt to directly learn which actions lead to desirable results without creating predictive models of the environment. In the example of a self-driving car, a model-based algorithm might attempt to physically model the friction between the tires and the road so it can predict how the vehicle's state evolves after an action, whereas a model-free algorithm simply learns which actions will take it to the goal.

Learning through a model-free approach can be more efficient, because often an agent does not need to be able to predict its environment fully in order to perform a task. Nonetheless, model-free learning does pose certain challenges. Because the goal is to maximize cumulative future rewards, the policy cannot be optimized for the immediate observed rewards alone. Furthermore, problems in reinforcement learning are typically stochastic and can have long horizons and large state-action spaces. Simply rolling out a policy and looking at cumulative rewards at the end of an episode results in very high variance estimates on how good the policy actually is, and it becomes difficult to pinpoint how the policy needs to change to improve its performance.

To alleviate the high variance of full rollouts, most modern methods attempt to learn a *value function* which models the expected future rewards (the value) that will be achieved from the current state onwards. The value function is learned by exploiting the fact that it should be temporally consistent. That is, the value of the current state should on average be equal to the immediate reward plus the value of the next state. The value of the next state can be considered a self-supervised label and is often called the target value throughout literature and this dissertation. How this target value is selected is a key design choice for a reinforcement learning algorithm. For example, picking the value of the action that the policy would have taken in the next state, leads to an algorithm that converges to the value of said policy. Choosing the action that maximizes the value function in turn converges to the optimal value function, which is the value of the optimal policy.

1.1.2. LIMITATIONS

Reinforcement learning has shown impressive results in domains with well-defined tasks and fast simulators. Notable successes include learning to play board games like Go and chess [Schrittwieser et al., 2020] or video games like Atari [Mnih et al., 2015] and StarCraft [Vinyals et al., 2019] at superhuman levels, without any prior human knowledge. More practical successes include the development of faster sorting and matrix multiplication algorithms [Mankowitz et al., 2023, Fawzi et al., 2022].

Unfortunately, to achieve these results reinforcement learning algorithms require interacting with the simulator or (video)game many times, often much more than a human would need to learn a task. This is prohibitive for many applications, particularly in settings where data is limited or expensive to collect. Going back to the self-driving car example, it is infeasible to train a reinforcement learning algorithm live on the road as it would simply take too long to gather the required experience, and also of course raise substantial safety concerns. Furthermore, even training in simulation is difficult since high-fidelity simulators will be expensive to run.

Addressing this limitation is an active area of research, and potentially has great consequences. Increasing the sample efficiency of reinforcement learning algorithms paves the way for more widespread application as well as reducing training costs.

1.1.3. EXPLORATION AND UNCERTAINTY

One reason these algorithms are inefficient is due to the problem statement itself. When learning from experience rather than labeled data, the agent only receives partial feedback. It can only experience the consequences of the action it takes, and has to essentially try different actions through trial and error to learn about the environment and task. Trying different actions for the sake of learning is called *exploration*, and is fundamental to the efficiency and performance of an algorithm. Too much exploration is wasteful and can hinder learning, whereas too little exploration can cause the agent to converge too quickly to suboptimal solutions.

An emerging direction in this context is the use of uncertainty-aware methods. Uncertainty in machine learning can be broadly divided into two types. Aleatoric uncertainty captures inherent randomness in the environment, such as noisy sensors or the previously mentioned stochastic transitions. This form of uncertainty cannot be reduced with more data. Epistemic uncertainty, on the other hand, reflects a lack of knowledge about the model or environment, and can be reduced through additional experience. For a self-driving car, the weather later that day can be considered aleatoric uncertainty, whereas the uncertainty over the tire traction in given weather conditions is epistemic uncertainty. Such epistemic uncertainty arises when the weather conditions were not encountered during training.

Particularly the *epistemic* uncertainty is of interest at training time to improve exploration. By quantifying the epistemic uncertainty about the value of actions or states, an agent can identify areas of the environment where its knowledge is limited. This signals the potential for learning new information, encouraging the agent to explore actions that may yield higher rewards but have not been sufficiently tried. Unlike naive exploration strategies, such as random action selection, uncertainty-driven exploration allows the agent to balance the trade-off between exploiting known rewarding actions and exploring unknown, potentially better options. Incorporating uncertainty estimates thus enables more efficient and targeted exploration, ultimately improving learning speed.

1.2. BAYESIAN METHODS

Bayesian methods provide a principled framework for reasoning under uncertainty by treating unknown quantities as random variables and updating beliefs in light of new

evidence. In contrast to point-estimate approaches, which commit to a single model or parameter value, Bayesian inference maintains a distribution over possible hypotheses, allowing for a natural representation of uncertainty. At the core of the Bayesian approach is Bayes' theorem, which describes how to update the posterior distribution over parameters given a prior belief, observed data and a likelihood function which specifies how likely a data point is given the model parameters. With a suitable prior and likelihood function, it is well known that Bayesian approaches perform well in statistical learning tasks.

Bayesian techniques are increasingly relevant in machine learning [Papamarkou et al., 2024], and reinforcement learning's sequential data collection fits naturally within the Bayesian paradigm, where the agent's uncertainty can guide exploration and each new observation in turn refines the posterior belief. Bayesian algorithms have extensively been studied in other decision problems such as bandits [Agrawal and Goyal, 2012, 2017, Russo and Van Roy, 2014, 2016], with theoretical guarantees on performance. Furthermore, Bayesian model-based reinforcement learning algorithms are also known to have good regret bounds [Osband et al., 2013] on smaller problems.

The strong theoretical guarantees and practical performance of existing Bayesian algorithms has motivated *Bayesian deep learning*, which combines Bayesian inference with neural networks [Neal, 1996]. The ability of neural networks to approximate high-dimensional functions together with the uncertainty quantification of Bayesian approaches has the potential of producing highly effective reinforcement learning methods. Unfortunately, there remain three main challenges in applying Bayesian deep learning to model-free reinforcement learning.

In deep learning in general, exact posterior inference is intractable, and the large parameter spaces of modern neural networks hinder the efficiency of existing approximate approaches. Developing approximate methods that are suitable to large neural networks is an active area of research with promising results [Chen et al., 2014, Wenzel et al., 2020, Garriga-Alonso and Fortuin, 2021]. Nonetheless, these methods are often more involved than traditional training methods that simply run an optimizer, and the high dimensionality makes it difficult to assess the quality of the approximation methods.

Another problem is caused specifically by learning from temporal differences. In supervised learning, the supplied label is assumed to be the ground truth with no uncertainty. However, in model-free reinforcement learning the target value is taken from the model itself, and therefore is uncertain. Properly propagating this uncertainty is essential for obtaining estimates that faithfully reflect the uncertainty at future time steps [Osband et al., 2018, O'Donoghue et al., 2018]. It is not immediately clear how to update the posterior appropriately while incorporating this uncertainty.

Finally, the motivating theoretical guarantees are contingent on correct assumptions on the likelihood and prior, but these are non-trivial to formulate for deep model-free reinforcement learning. The complexity and large parameter spaces of modern neural networks make it unclear how priors over parameter spaces translate to predictive distributions, and the likelihoods assumed on temporal difference errors are never guaranteed to be true by the definition of MDPs [Dearden et al., 1998]. This is less of an issue in model-based reinforcement learning where the definition of an MDP specifies an exact generative model of the environment [Osband et al., 2013].

It is *possible* to ignore these issues, and existing work on Bayesian deep model-free reinforcement learning has focused on the first problem. For example Dearden et al. [1998], Azizzadenesheli et al. [2018] and Schmitt et al. [2023] propose model classes that are easier to perform inference over, and Fortunato et al. [2019], Dwaracherla and Roy [2021] and Ishfaq et al. [2023] use sampling based approaches to approximate posteriors. These solutions are not widely used in practice however, as they often do not outperform simple ensembles and are more difficult to implement and tune.

Summarizing, theory on simplified problems has motivated Bayesian deep learning approaches. Inference and priors are problems that are general to deep learning, and updating the posterior and likelihoods are problems caused by model-free reinforcement learning specifically. Most recent Bayesian deep reinforcement learning literature has focused on inference.

1.3. THEORETICAL MOTIVATIONS IN DEEP LEARNING

This dissertation attempts to strike a middle ground between theory and pure practice. While attempts at fully understanding how neural networks learn exist, it is an overall trend in the field that neural networks are black boxes, and methods often lack rigorous theoretical guarantees. Nonetheless, algorithms, design choices and heuristics are often *theoretically motivated* by derivations or theorems in simplified cases. For example, while deep Q-learning does not have any theoretical guarantees to converge, the loss function and overall algorithm structure is derived from Q-learning, which *is* known to converge under assumptions. Many other algorithms in the field follow this pattern. Rainbow [Hessel et al., 2018] builds upon deep Q-learning and includes theoretically motivated design choices such as double Q-learning [Van Hasselt, 2010], and the clipping heuristic in Proximal Policy Optimization is motivated as a simpler version of a more formal trust-region approach [Schulman et al., 2017].

Likewise, Bayesian model-free deep reinforcement learning algorithms are motivated by the utility of the Bayesian paradigm in other settings, even though theory does not directly translate. Deep learning research often combines theoretical motivation as to why something can be expected (not guaranteed) to work with experimental results in order to fully motivate an algorithm or method. In this dissertation I take the same approach and motivate design choices with theory, while testing experimental performance.

1.4. RESEARCH QUESTIONS

The goal of this dissertation is to shed light on if Bayesian model-free methods should be expected to work well, both in theory and practice. This leads us to formulate the following main research question of this dissertation: *Can Bayesian model-free methods be expected to achieve higher cumulative return in the same number of training samples than their non-Bayesian counterparts?* We answer the main research question through three sub-questions:

1. Can ensembles be viewed as a Bayesian approximation to the posterior?
This subquestion aims to relate Bayesian algorithms to ensembles, which are a very common and highly performant uncertainty quantification method in reinforcement learning. Ensemble-based reinforcement learning algorithms are sometimes motivated from

a Bayesian point of view [Osband et al., 2018]. However, in practice these methods simply train multiple models in parallel, relying on initialization randomness to produce a diverse set of models. The main intuition is that this initialization randomness will cause the different models in the ensemble to disagree on data points they were not trained on, giving a sense of uncertainty. In practice, averaging ensemble models is known to improve performance [Lakshminarayanan et al., 2017], and Bayesian model averaging is known theoretically to outperform a single model. With this question we aim to connect the performance of ensembles to Bayesian theory.

2. Can the general structure of deep Bayesian model-free reinforcement learning methods be expected to converge?
There are several existing practical implementations of deep Bayesian model-free algorithms [Osband et al., 2018, Azizzadenesheli et al., 2018, Dwaracherla and Roy, 2021, Ishfaq et al., 2023, Schmitt et al., 2023]. These algorithms typically differ in posterior approximation methods, but also seemingly make different design decisions on selecting their target values. Non-Bayesian deep reinforcement learning algorithms are typically motivated by the fact their model updates are guaranteed to converge in tabular settings, i.e., without neural networks. With this question we aim to make a similar convergence argument for deep Bayesian model-free reinforcement learning, and at the same time unify seemingly different algorithms into a general algorithm.

3. Are the commonly assumed likelihoods and priors realistic in typical benchmark tasks, and can they be improved?
Bayesian algorithms are defined by two central assumptions: the likelihood distribution and the prior. Typically, it is important that these assumptions are in line with reality in order for the Bayesian posterior distribution to accurately reflect our knowledge. However, typically Bayesian deep model-free reinforcement learning algorithms assume simple Gaussian distributions as likelihood and prior [Dearden et al., 1998, Osband et al., 2018, Azizzadenesheli et al., 2018, Dwaracherla and Roy, 2021, Ishfaq et al., 2023, Schmitt et al., 2023], both out of simplicity and as a natural extension of mean squared error and common weight decay. With this research question we aim to check how aligned assumptions are with reality, and if more accurate assumptions lead to more performant algorithms.

1.5. CONTRIBUTIONS

In this dissertation, we explore how Bayesian methods can aid exploration in model-free reinforcement learning, both from a practical and a theoretical point of view. The main contributions of this dissertation are the following:

1. A Bayesian point of view on training ensembles of Q-learning agents. We draw a connection between deep ensembles and Bayesian posterior inference by proposing a novel variant of Deep Q-Learning (DQN) that uses a sequential Monte Carlo sampler to approximate the posterior over Q-functions. The resulting set of models can be used in the same manner as standard ensembles, while being grounded in a principled Bayesian interpretation. We empirically test our approach and observe that our agents are capable of efficient exploration.

2. A unifying framework and convergence analysis for Bayesian model-free reinforcement learning approaches. We study existing Bayesian deep Q-learning algo-

rithms in a simplified setting and unify them within a single general framework. We prove that this generalized approach is guaranteed to converge under mild assumptions, providing theoretical support for a family of existing algorithms. Building on these insights, we derive a new uncertainty-aware variant of a popular model-free RL algorithm that we experimentally test on a variety of benchmark tasks.

3. An analysis of misspecification of likelihoods and priors for Bayesian deep Q-learning. Through extensive experiments, we statistically test the validity of the Gaussian likelihood and prior assumptions commonly made in Bayesian model-free algorithms, and show that they are consistently violated on a variety of benchmark tasks. We propose Laplace priors as a simple drop-in replacement that is empirically closer to the true distribution of parameters and improves performance on several benchmarks. We go one step further by introducing a meta-learned prior fitted on a set of training tasks, and show that it generalizes to unseen tasks, yielding superior performance to both Gaussian and Laplace priors.

1.6. OUTLINE

The remainder of this dissertation is organized as follows. Chapter 2 addresses the first research question, drawing connections between ensembles of models and Bayesian inference through a Sequential Monte Carlo variant of DQN. Chapter 3 addresses the second research question by unifying existing Bayesian Q-learning algorithms into a single framework with convergence guarantees in simplified settings, and leveraging these insights to design a new uncertainty-aware method. Chapter 4 addresses the third research question by empirically examining the likelihood and prior assumptions underlying Bayesian model-free RL, and proposing improved alternatives. Finally, Chapter 5 summarizes the findings of this dissertation and discusses several promising directions for future research.

2

BAYESIAN ENSEMBLES FOR EXPLORATION IN DEEP Q-LEARNING

Bayesian algorithms perform well in statistical learning, and theoretical results exist for Bayesian reinforcement learning in tabular settings. In deep learning however, approximating the posterior is non-trivial. On the other hand, ensemble techniques perform quite well and are simple to implement. Can we combine the practicality of ensembles with the theory of the Bayesian paradigm?

In Section 2.3 we propose to employ Sequential Monte Carlo samplers to sample from the posterior distribution, effectively "training"' an ensemble. In Section 2.4 we propose to use this model in a standard Q-learning agent to maintain a posterior over Q-values. We show the effectiveness of our method experimentally in Section 2.5.

This chapter is based on Pascal R. van der Vaart, Neil Yorke-Smith and Matthijs T. J Spaan. "Bayesian Ensembles for Exploration in Deep Q-Learning" in *In Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, 2024

2.1. INTRODUCTION

Reinforcement learning (RL) algorithms are still notoriously sample inefficient. One pressing reason is the difficulty of exploring an environment efficiently while assuming little prior knowledge [Yang et al., 2021]. A promising approach that is currently studied is to quantify uncertainty in the value models learned by the agent, and then either provide intrinsic reward, be optimistic, or use Thompson sampling to explore [Bellemare et al., 2016, Ostrovski et al., 2017, Burda et al., 2018, Osband et al., 2016, 2018, Gal and Ghahramani, 2016, Fortunato et al., 2019, Azizzadenesheli et al., 2018, Ciosek et al., 2019, Lee et al., 2021, Bai et al., 2021]. However, quantifying uncertainty for deep neural networks is in itself a difficult task [Hüllermeier and Waegeman, 2021, Lockwood and Si, 2022].

Ensembles of neural networks have been shown to provide better predictive accuracy over a single model in supervised learning tasks [Dietterich, 2000, Lakshminarayanan et al., 2017], as well as suitable methods for uncertainty quantification for exploration in reinforcement learning [Osband et al., 2016, 2018, Fellows et al., 2021]. While ensembles with independent models of identical architecture tend to collapse to the same predictive model [Geiger et al., 2020], there are several techniques developed to prevent this, such as adversarial learning [Lakshminarayanan et al., 2017], bootstrapping the data [Osband et al., 2016], and adding additive priors [Osband et al., 2018]. Further, some techniques such as Stein Variational Gradient Descent [Liu and Wang, 2016, D'Angelo and Fortuin, 2021] alleviate this issue by interpreting the ensemble as an approximation to the Bayesian posterior and training it as such. The method that we propose falls into this last category and aims to be closer to the posterior for more accurate uncertainty quantification, while retaining the flexibility of ensembles.

Bayesian neural networks can have desirable properties if the posterior can be inferred accurately. They have in theory optimal predictive accuracy given the correct likelihood and prior and also provide accurate uncertainty quantification. Unfortunately, exactly inferring the posterior is intractable already for some simple statistical models, and accurately approximating this posterior is very difficult for neural networks. Typically, posterior approximation methods fall into one of two categories: Markov Chain Monte Carlo (MCMC), and Variational Inference.

Recently both types of methods have been altered specifically for application to neural networks. Variational Inference can be scaled to large networks by picking simple model classes to tractably optimize within the class and has been applied to RL in the context of uncertainty quantification and exploration [Gal and Ghahramani, 2016, Fortunato et al., 2019, Schmitt et al., 2023]. Due to the complex nature of Neural Networks however, it is unclear how the model class biases uncertainty quantification. On the other hand, MCMC is in theory unbiased and also shows strong results in large networks in practice [Chen et al., 2014, Wenzel et al., 2020, Garriga-Alonso and Fortuin, 2021]. However, for complex multimodal distributions, MCMC methods can struggle to find every mode [Del Moral et al., 2006]. This is an important drawback in deep learning, where the posterior distribution is likely very ill behaved, and especially in RL where under-approximation of the posterior complexity might lead to underestimating the uncertainty and therefore failure of exploration. Sequential Monte Carlo (SMC), which uses a set of particles to approximate the posterior, can be a remedy to these issues in non-

deep learning applications [Del Moral et al., 2006].

In this work, we forego Variational Inference to avoid a decision in model class, and instead alleviate the issues in MCMC by using SMC. Noting the success of ensembles in deep learning, we unify ensembles and MCMC methods by using SMC algorithms to train an ensemble in a Bayesian manner, to benefit from both the practical effectiveness of ensembles and theoretical foundations of MCMC. Specifically, our contributions are as follows:

1. We adapt existing SMC algorithms to a minibatch setting, and show empirically that they are feasible methods to train ensembles so that they serve as proper approximations to the Bayes posterior.

2. As our main contribution, we introduce Sequential Monte Carlo DQN (SMC-DQN)[1], an RL algorithm which uses SMC to track a posterior over the Q-values in a theoretically sound manner, by sequentially updating its models with incoming data and correctly conditioning on target parameters. The agent uses Thompson sampling from the posterior distribution to drive exploration.

3. We experimentally test our agent's exploration capabilities on several environments, observing significantly stronger performance over regular ensembles and results that are competitive with a strong baseline.

2.2. BACKGROUND

In this section we introduce the necessary background for our contribution.

2.2.1. MARKOV DECISION PROCESSES

Let $\Delta(X)$ denote the set of probability distributions over the set X. A Markov Decision Process is a tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$ of a state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and discount factor $0 \leq \gamma < 1$. At each time step t, an agent observes the current state s_t, chooses an action $a_t \sim \pi(s_t)$ according to its policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, and receives reward $r_t = R(s_t, a_t)$. The goal of reinforcement learning is to find a policy π that maximizes the discounted cumulative reward $\mathbb{E}_{T,\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$. Of central importance is the Q-function

$$Q^{\pi}(s,a) = R(s,a) + \mathbb{E}_{T,\pi}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a\right],$$

denoting the expected discounted future reward if the agent executes action a in state s and then follows the policy π.

Since the transition function T and reward function R are assumed to be unknown to the agent, computing a strong policy requires exploration of the environment to learn which actions result in optimal return.

[1] Code is available at `https://github.com/Pascal314/BayesianEnsemblesAAMAS`

2.2.2. DEEP Q-LEARNING

A common strategy to find an optimal policy is Deep Q-learning (DQN) [Mnih et al., 2015], where a parameterized neural network Q_θ is trained to minimize the temporal difference error

$$\arg\min_\theta \left[Q_\theta(s,a) - r - \gamma \max_{a'} Q_{\theta'}(s',a') \right]^2, \tag{2.1}$$

with (s,a,r,s') sampled from the environment or a replay buffer, and θ' are the target parameters. $Q'_\theta(s',a')$ is known as the *target value*, which can be thought of what we believe is true in the next state. The target parameters θ' are periodically updated $\theta' \leftarrow \theta$ at a slower pace than θ. This keeps the regression target for $Q_\theta(s,a)$ more stable. For more details we refer to Mnih et al. [2015]. The corresponding policy picks $\arg\max_a Q_\theta(s,a)$ in every state. The agent can pick a random action instead with probability ϵ, to perform exploratory actions. However, such random exploration can be inefficient.

2.2.3. THOMPSON SAMPLING

Rather than taking uniformly random actions to explore, Thompson sampling [Thompson, 1933] picks actions according to their posterior probability of being optimal. Originally proposed as an algorithm for multi armed bandits, Thompson sampling samples a model $Q_\theta \sim p(\theta|\mathcal{D})$ and acts greedily with respect to that model $a = \arg\max Q(s,a)$ for a time step or full episode. Thompson sampling is well studied in the bandit and model-based reinforcement learning settings, and known to have good regret bounds [Agrawal and Goyal, 2012, 2017, Russo and Van Roy, 2014, Osband et al., 2013]. This has served as motivation to apply Thompson sampling to deep model-free reinforcement learning as well.

2.2.4. BOOTSTRAPPED DQN

The BootDQN algorithm [Osband et al., 2016] implements a crude form of Thompson sampling by leveraging the uncertainty estimation abilities of ensembles. By learning an ensemble of Q-networks

$$Q_{\theta_1}(s,a), \ldots, Q_{\theta_n}(s,a)$$

the agent achieves deep exploration through sampling uniformly $i \in \{1,\ldots,n\}$ and acting greedily with respect to the network Q_{θ_i} for a full episode by picking $\arg\max_a Q_{\theta_i}(s,a)$ in every state s. To update its predictions, each network Q_{θ_i} is equipped with its own target network $Q_{\theta'_i}$, and gets updated with its own targets:

$$\theta_i \leftarrow \theta_i - \alpha \nabla_{\theta_i} \left[Q_{\theta_i}(s,a) - r - \max_{a'} Q_{\theta'_i}(s',a') \right]^2, \tag{2.2}$$

where (s,a,r,s') are transitions sampled uniformly from a replay buffer and α is a learning rate.

Crucially, the ensemble $Q_{\theta_1}(s,a), \ldots, Q_{\theta_n}(s,a)$ has to stay diverse in under-explored states in order to keep exploration going. This can be achieved by bootstrapping the data for each ensemble member, or by using randomized prior functions. Randomized prior functions have been shown to be effective at keeping ensemble diversity [Osband et al., 2018].

A randomized prior function is a fixed function $Q_{\vartheta_i}(s,a)$ that is sampled independently for each ensemble member $Q_{\theta_i}(s,a)$, at the start of training. It remains unmodified during training and is added to the model outputs:

$$(Q_{\theta_i} + Q_{\vartheta_i})(s,a) = Q_{\theta_i}(s,a) + Q_{\vartheta_i}(s,a).$$

During action selection, the Q-values of ensemble member i are given by $(Q_{\theta_i}+Q_{\vartheta_i})(s,a)$, and the BootDQN update (2.2) is modified to the following:

$$\theta_i \leftarrow \theta_i - \alpha \nabla_{\theta_i} \left[(Q_{\theta_i} + Q_{\vartheta_i})(s,a) - r - \max_{a'} (Q_{\theta'_i} + Q_{\vartheta_i})(s',a') \right]^2. \tag{2.3}$$

The fact that each ensemble member has a unique prior function causes unique generalization behaviour on unobserved data. This keeps the ensemble outputs diverse in under-explored states.

However, while BootDQN is motivated as an approximation to Thompson sampling, randomized prior functions lack theoretical motivation when considered as Bayesian priors for neural networks. In problems with well-defined likelihoods and priors, the Bayesian posterior can therefore be expected to outperform methods that rely on randomized prior functions.

2.2.5. BAYESIAN NEURAL NETWORKS

A Bayesian Neural Network (BNN) is any neural network f_θ parameterized by $\theta \in \Theta$, with some prior distribution $p(\theta)$ over Θ. Given training data $(x_1,\dots,x_n)$ and labels $(y_1,\dots,y_n)$ i.i.d. from some likelihood $\mathscr{L}(y|f_\theta(x))$, the goal is to compute or sample from the posterior distribution over the parameter θ:

$$\begin{aligned} p(\theta|\mathscr{D}) &= \frac{p(\mathscr{D}|\theta)\,p(\theta)}{\int p(\mathscr{D}|\theta)\,p(\theta)\,d\theta} \\ &= \frac{\prod_{i=1}^n \mathscr{L}(y_i|f_\theta(x_i))\,p(\theta)}{\int \prod_{i=1}^n \mathscr{L}(y_i|f_\theta(x_i))\,p(\theta)\,d\theta}. \end{aligned} \tag{2.4}$$

Unfortunately, especially in the case of large neural networks, the posterior is intractable to compute or sample from exactly. Therefore it is necessary to resort to approximation methods such as variational inference or MCMC.

2.2.6. SEQUENTIAL MONTE CARLO

Sequential Monte Carlo algorithms model a sequence of distributions $p_0(\theta),\dots,p_m(\theta)$. An initial state of particles is drawn from p_0, and by repeatedly applying importance sampling, an MCMC algorithm and resampling steps, the initial samples are transformed to be a sample of $p_m(\theta)$. A basic outline is given in Algorithm 1. Under certain conditions, notably that sample distributions have to be invariant under the MCMC steps, this Monte Carlo scheme converges to the correct target [Del Moral et al., 2006].

Leveraging this fact, we can set

$$p_m(\theta) = p(\theta|\mathscr{D}) \propto p(\theta)\,p(\mathscr{D}|\theta), \tag{2.5}$$

Algorithm 1: Base Sequential Monte Carlo

Input: sequence of target distributions $p_0(\theta), \ldots, p_m(\theta)$ and MCMC kernels $P_1, \ldots P_m$
Result: a sample $\theta_1, \ldots, \theta_n \sim p_m(\theta)$
$\theta_1, \ldots, \theta_n \sim p_0$
$w_1, \ldots, w_n \leftarrow 1$
for $t = 1, \ldots, m$ **do**
  $\theta_1, \ldots, \theta_n \sim \texttt{Categorical}(\{\theta_i, w_i\}_{i=1}^n)$
  for $j = 1, \ldots n$ **do**
    $w_j \leftarrow \frac{p_t(\theta_j)}{p_{t-1}(\theta_j)}$
  end
  for $j = 1, \ldots n$ **do**
    $\theta_j \leftarrow P_t(\theta_j)$
  end
end

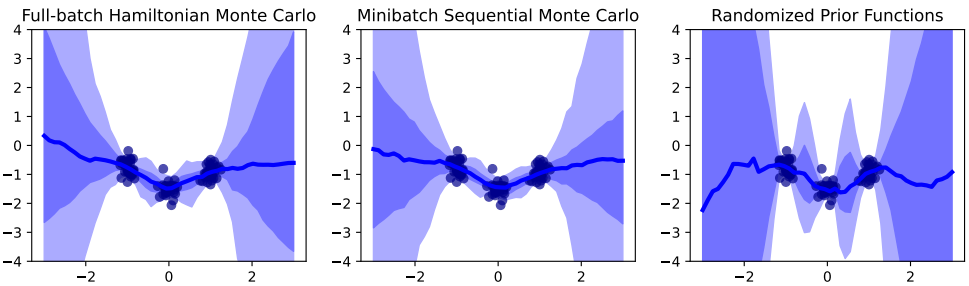

Figure 2.1: The 25th-75th and 5th-95th quantiles of the predictive posterior as predicted by SMC and an ensemble with randomized prior functions in a supervised learning setting, compared to the gold standard Hamiltonian Monte Carlo without noise. Sequential Monte Carlo more closely resembles the posterior distribution as approximated by Hamiltonian MC.

pick a sequence of temperatures

$$0 = \lambda_0 < \lambda_1 < \cdots < \lambda_m = 1,$$

and use SMC to sample from $p_0(\theta), \ldots, p_m(\theta)$, where the distributions are given by

$$p_t(\theta) \propto p(\mathscr{D}|\theta)^{\lambda_t} p(\theta). \tag{2.6}$$

Equation 2.6 effectively interpolates between the prior and the posterior. Setting the temperature sequence correctly is important, because a too coarse interpolation can cause the importance sampling weights to be unstable and a too fine interpolation wastes computation. Fortunately, automatic on-the-fly tuning methods exist that choose the next temperature based on the effective sample size of the current sample [Cai et al., 2021, Dau and Chopin, 2022].

2.3. SEQUENTIAL MONTE CARLO FOR BNNS

Having introduced the necessary background, to prepare for our main contribution, we next analyze SMC in the context of Bayesian Neural Networks.

Applying SMC to model the posterior of a Bayesian Neural Network is in practice similar to an ensemble. The particles $\theta_1, \ldots, \theta_n$ are individual models, and equipped with importance sampling weights $w_1, \ldots, w_n$ model a theoretically unbiased approximation of the predictive posterior distribution.

The model is initialized by sampling initial parameters $\theta_1, \ldots, \theta_n$ from the prior $p(\theta)$, and trained by running SMC with target distributions $\left(p(\mathscr{D}|\theta)^{\lambda_t} p(\theta)\right)_{t \geq 0}$. Unfortunately, typically in deep learning the data set $\mathscr{D}$ is so large that mini-batches are required to tractably compute the likelihood and gradients. This poses two problems:

1. The reweighting step is now noisy, which lowers the quality of importance sampling.

2. The MCMC step, which typically is gradient based, is now noisy. This means we may have to rely on MCMC methods that only approximately leave the target distribution invariant.

However, these problems can largely be alleviated, as the theoretical results by Llorente et al. [2022] show that noisy importance sampling, which is central to the SMC algorithm, has higher variance but remains unbiased. Furthermore, Wenzel et al. [2020] and Garriga-Alonso and Fortuin [2021] show that accurate noisy MCMC kernels that leave the target distribution (approximately) invariant do exist, so we can use these kernels specifically crafted for mini-batch noise in our SMC algorithm.

Specifically, in our experiments we estimate the reweighting steps by sampling a single batch independently for every particle every iteration. As MCMC kernel we use the Symplectic Euler Langevin scheme with hyper-parameters as suggested by Wenzel et al. [2020]. At each iteration t, the temperature in the next step $\lambda_{t+1} = \lambda_t + \delta$ is picked by keeping the effective sample size (ESS) at a desired level d:

$$\begin{aligned} &\max_{\delta} \delta, \text{ such that: } \delta < 1 - \lambda_t, \text{ and} \\ &\text{ESS} = \frac{\left(\sum_{j=1}^n w_j\right)^2}{\sum_{j=1}^n w_j^2} = \frac{\left(\sum_{j=1}^n p(\mathscr{D}|\theta_j)^{\delta}\right)^2}{\sum_{j=1}^n p(\mathscr{D}|\theta_j)^{2\delta}} > d. \end{aligned} \tag{2.7}$$

Our initial experimental findings in a supervised learning setting, shown in Figure 2.1, show a comparison of the predictive posteriors approximated by noise-free Hamiltonian Monte Carlo, Mini-batch Sequential Monte Carlo, and an ensemble with randomized prior functions. Even with mini-batches, SMC with noisy likelihoods and gradients results in performance on par with the gold standard noise-free Hamiltonian Monte Carlo [Neal et al., 2011]. However, small mini-batches may require a finer grained interpolation, since the temperature schedule depends on maintaining a high enough ESS, which is known to be lower when using noisy reweighting [Llorente et al., 2022].

Algorithm 2: Sequential Monte Carlo for BNNs

Input: Prior $p_0(\theta)$ and target $p_0(\theta)\,p(\mathscr{D}|\theta)$, MCMC algorithm
Result: A sample $\theta_1,\dots,\theta_n \sim p_0(\theta)\,p(\theta|\mathscr{D})$
$\theta_1,\dots,\theta_n \sim p_0$
$w_1,\dots,w_n \leftarrow 1$
$t \leftarrow 0$
$\lambda_0 \leftarrow 0$
while $\lambda_t < 1$ **do**
  Pick $\lambda_t > \lambda_{t-1}$ (eq. 2.7)
  $\log p_t(\theta) \leftarrow \log p_0(\theta) + \lambda_t \log p(\mathscr{D}|\theta)$
  $\theta_1,\dots,\theta_n \sim \texttt{Categorical}(\{\theta_i, w_i\}_{i=1}^n)$
  for $j = 1,\dots n$ **do**
    $\log w_j \leftarrow (\lambda_t - \lambda_{t-1}) \log p(\theta_j|\mathscr{D})$
  end
  $w_1,\dots,w_n = \texttt{normalize}(w_1,\dots,w_n)$
  for $j = 1,\dots n$ **do**
    $\theta_j \leftarrow \texttt{MCMC}(\theta_j, \log p_t(\theta))$
  end
  $t \leftarrow t+1$
end

2.4. SEQUENTIAL MONTE CARLO DQN (SMC-DQN)

With the goal of improving exploration, we construct an agent that can accurately quantify uncertainty in its Q-values by approximating the posterior distribution over its parameters given the transitions that have previously been observed. Specifically, we extend a standard DQN agent by replacing its point-wise estimator $Q_\theta(s,a)$ with an ensemble of neural networks $Q_{\theta_1}(s,a),\dots,Q_{\theta_n}(s,a)$ and weights $w_1,\dots w_n$ to maintain an approximation of the posterior $p(\theta|\mathscr{D},\boldsymbol{\theta}')$, conditioned on the current replay buffer

$$\mathscr{D} = ((s_t, a_t, r_t, s_{t+1}))_{t=1\dots N}$$

and current target parameters $\boldsymbol{\theta}' = (\theta'_1,\dots,\theta'_n)$. In line with the work by Schmitt et al. [2023], a normal distribution

$$Q_\theta(s,a) - r(s,a) - \gamma \max_{a'} Q_{\theta'}(s',a') \sim \mathscr{N}(0,\sigma^2)$$

is used as a probabilistic interpretation of the squared temporal difference error, and to represent the uncertainty in the targets we define the likelihood to be a mixture distribution

$$\log \mathscr{L}(s,a,r,s'|\theta,\boldsymbol{\theta}') = \log \sum_{i=1}^{n} \frac{1}{n} \exp\left(\frac{1}{2\sigma^2}[Q_\theta(s,a) - r(s,a) - \gamma \max_{a'} Q_{\theta'_i}(s',a')]^2\right), \tag{2.8}$$

contrasting BootDQN which shares no target values between ensemble members. The log posterior distribution is defined as

$$\log p(\theta|\boldsymbol{\theta}',\mathscr{D}) \propto \log p(\theta) + \log \mathscr{L}(\mathscr{D}|\theta,\boldsymbol{\theta}'), \tag{2.9}$$

where

$$\log \mathscr{L}(\mathscr{D}|\theta,\boldsymbol{\theta}') = \sum_{(s,a,r,s')\in\mathscr{D}} \log \mathscr{L}(s,a,r,s'|\theta,\boldsymbol{\theta}'). \tag{2.10}$$

After collecting a new batch of trajectories

$$\mathscr{B} = ((s_t, a_t, r_t, s_{t+1}))_{t=1,\dots,B}$$

by acting in the environment, the posterior distribution can be updated efficiently by noting that

$$\begin{aligned} &\log p(\theta|\boldsymbol{\theta}',\mathscr{D}\cup\mathscr{B}) \\ &\qquad = \log p(\theta) + \log \mathscr{L}(\mathscr{D}\cup\mathscr{B}|\theta,\boldsymbol{\theta}') \\ &\qquad = \log p(\theta) + \log \mathscr{L}(\mathscr{D}|\theta,\boldsymbol{\theta}') + \log \mathscr{L}(\mathscr{B}|\theta,\boldsymbol{\theta}') \\ &\qquad = \log p(\theta|\boldsymbol{\theta}',\mathscr{D}) + \log \mathscr{L}(\mathscr{B}|\theta,\boldsymbol{\theta}'). \end{aligned} \tag{2.11}$$

Therefore, since our agent is currently holding a sample of $p(\theta|\boldsymbol{\theta}',\mathscr{D})$, the posterior can be updated by running SMC on the sequence

$$p(\theta|\boldsymbol{\theta}',\mathscr{D}),\ p(\theta|\boldsymbol{\theta}',\mathscr{D})\mathscr{L}(\mathscr{B}|\theta,\boldsymbol{\theta}')^{\lambda_1},\ \dots, \tag{2.12}$$

$$p(\theta|\boldsymbol{\theta}',\mathscr{D})\mathscr{L}(\mathscr{B}|\theta,\boldsymbol{\theta}')^{\lambda_k},\ p(\theta|\boldsymbol{\theta}',\mathscr{D})\mathscr{L}(\mathscr{B}|\theta,\boldsymbol{\theta}'), \tag{2.13}$$

which interpolates between the posterior given what was already known, and the new posterior including the new batch. This is equivalent to Equation 2.6 when taking $p(\theta|\boldsymbol{\theta}',\mathscr{D})$ as prior and $p(\mathscr{B}|\theta,\boldsymbol{\theta}')$ as likelihood.

To evaluate Equation 2.12, the likelihood $\mathscr{L}(\mathscr{B}|\theta,\boldsymbol{\theta}')$ only requires the latest batch, and can easily be computed exactly. However, we choose to approximate $p(\theta|\boldsymbol{\theta}',\mathscr{D})$ by sampling mini-batches uniformly from the replay buffer, since computing it exactly would require summing over the entire replay buffer.

Updating the target networks $\boldsymbol{\theta}'$ changes the target distribution, meaning that the sample

$$\theta_1,\dots,\theta_n,\ w_1,\dots,w_n$$

is no longer a sample of the posterior with respect to the updated targets, i.e., $p(\theta|\boldsymbol{\theta}'_{\texttt{new}},\mathscr{D})$. Therefore, the typical target update $\theta'_i \leftarrow \theta_i$ is now accompanied by another SMC step, which smoothly interpolates between the distribution conditioned on the old targets and the distribution conditioned on the new targets via the sequence

$$p(\theta)\mathscr{L}(\mathscr{D}|\theta,\boldsymbol{\theta}'_{\texttt{old}})^{(1-\lambda_t)}\, p(\theta)\mathscr{L}(\mathscr{D}|\theta,\boldsymbol{\theta}'_{\texttt{new}})^{\lambda_t} \to p(\theta)\mathscr{L}(\mathscr{D}|\theta,\boldsymbol{\theta}'_{\texttt{new}}), \tag{2.14}$$

as λ_t increases from 0 to 1. This transforms a sample of the posterior with respect to the previous targets to a sample with respect to the new targets. Intuitively speaking,

Algorithm 3: SMC-DQN

Input: MDP, batch size B, ensemble size n
Result: Posterior over $Q_\theta(s,a)$
$\theta_1,\dots,\theta_n \sim p(\theta)$
$\theta'_1,\dots,\theta'_n \leftarrow \theta_1,\dots,\theta_n$
$w_1,\dots,w_n \leftarrow 1$
while *training* **do**
  $i \sim \texttt{uniform}(1,\dots,n)$
  $s_0 \sim \texttt{MDP}$
  $t \leftarrow 0$
  while *episode not done* **do**
    $a_t = \arg\max_a Q_{\theta_i}(s_t)$
    $r_t, s_{t+1} \sim \texttt{MDP}(s_t, a_t)$
    $\mathscr{B} = \mathscr{B} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
    if $|\mathscr{B}| = B$ **then**
      $\theta_1,\dots,\theta_n, w_1,\dots,w_n \leftarrow \texttt{SMC}(p(\theta|\boldsymbol{\theta}', \mathscr{D}), p(\theta|\boldsymbol{\theta}', \mathscr{D} \cup \mathscr{B}))$ (eq. 2.12)
      $\mathscr{D} \leftarrow \mathscr{D} \cup \mathscr{B}$
      $\mathscr{B} \leftarrow \emptyset$
    end
    if *time for target update* **then**
      $\theta'_{i,\texttt{new}} \leftarrow \theta_i$
      $\theta_1,\dots,\theta_n, w_1,\dots,w_n \leftarrow \texttt{SMC}(p(\theta|\boldsymbol{\theta}', \mathscr{D}), p(\theta|\boldsymbol{\theta}'_{\texttt{new}}, \mathscr{D}))$ (eq. 2.14)
      $\theta'_i \leftarrow \theta'_{i,\texttt{new}}$
    end
  end
end

this trains the main networks $\theta_1,\dots,\theta_n$ to match the new targets, before acting in the environment again.

Similarly to BootDQN, actions are selected by Thompson sampling one model θ_i, and committing to this model for a full episode. Algorithm 3 shows the general structure with both updates, where SMC refers to Algorithm 2. The Symplectic Euler Langevin scheme [Wenzel et al., 2020] is used as MCMC step. We opt not to resample in every SMC step, in order to maintain more diversity in the ensemble. Further, to maximize the exploration behaviour of the agent, we sample uniformly from the particles as opposed to utilizing the weights.

While the structure is similar to a typical DQN implementation, a major difference is that the SMC steps take significantly more time than typical Q-value updates. For each batch from the environment, the agent adapts its model to match the posterior given the new replay buffer, and also after each target update the agent trains its networks to match the posterior given the new targets. The speed at which the agent can condition its model on the new data causes a trade off between computation complexity and sample complexity. In this work we are mostly concerned with sample complexity in the

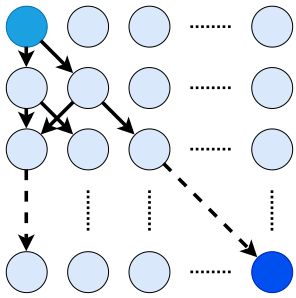

Figure 2.2: Graphical representation of Deep Sea. The agent starts in the top left state, and transitions downwards to the left or right at every time step depending on the action taken. The only positive reward is granted in the bottom right state

exploration setting. Furthermore, there is a very clear split between samples that are already in the data set $\mathscr{D}$ and samples that have yet to be incorporated $\mathscr{B}$. It is assumed that the replay buffer $\mathscr{D}$ is large enough to store every experienced transition.

2.5. EXPERIMENTAL STUDY

So far we have introduced SMC-DQN, an agent that is architecturally similar to DQN with an ensemble, but employs SMC to maintain a posterior over Q-value functions. To test our agent's ability to explore in difficult sparse reward environments, we evaluate on the exploration tasks in Behaviour Suite (BSuite) [Osband et al., 2020]. We also test on BSuite's Mountain Car environment, which requires further exploration after reaching the goal state to recover an optimal policy, to test whether our agent still explores effectively after finding reward.

2.5.1. ENVIRONMENTS

Our agents run for 10000 episodes on both Deep Sea environments, 1000 episodes on Cartpole, and 300 on Mountain Car, which, except for in Mountain Car, is the number of episodes prescribed by BSuite. We only run our agents for 300 episodes as opposed to BSuite's 1000 episodes on Mountain Car because performance plateaus already after 100 episodes.

Deep Sea Deep Sea is a one hot encoded chain-like environment, where the observations are a matrix of size $n \times n$ and there are two discrete actions. The agent starts at the top left state, moves down one row at every time step, and also moves left or right depending on the action taken. A graphical view is shown in Figure 2.2. Only the bottom right state results in positive reward, meaning that the agent has to execute the correct action n times in a row to observe any positive reward in an episode; hence this environment is impossible to solve with an ϵ-greedy approach. Further, the agent receives a small negative reward when moving to the right. In our experiments, we use an environment of size 30×30 and report the episodic return excluding the small negative reward, so an optimal policy receives reward 1 in every episode. This is a difficult exploration

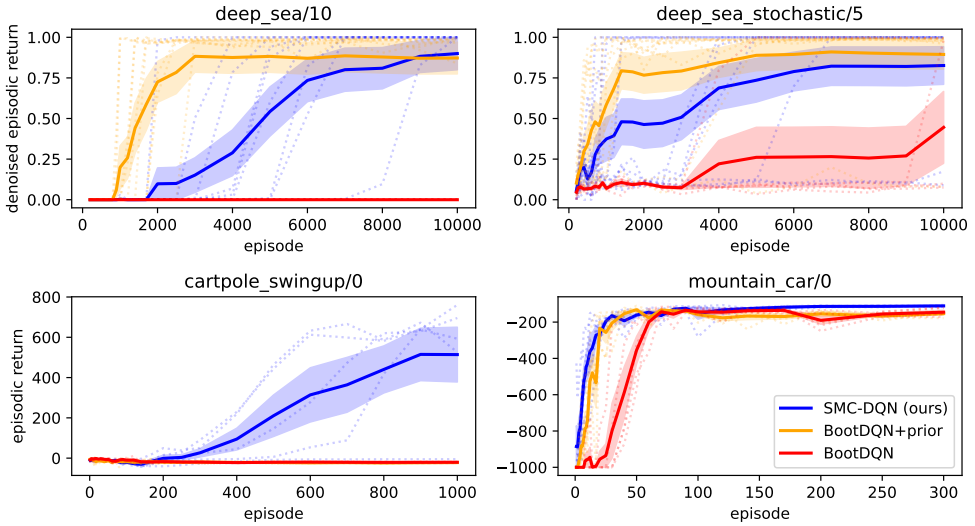

Figure 2.3: Learning curves over the BSuite environments. The solid line is the mean of 10 seeds for the Deep Sea environments, and 5 seeds for Cartpole Swingup and Mountain Car. The shaded area denotes the standard error of the mean. Dashed lines show individual seeds.

task where the agent can not rely on generalization. In both deep sea variants, which action goes left or right is randomized for each state to avoid trivial solutions.

Deep Sea Stochastic Stochastic Deep Sea adds a $(1-\frac{1}{n})$ probability that the 'right' action fails and goes to the left instead. Furthermore, on the bottom left state the agent receives a stochastic reward with mean 0. We use an environment of size 20×20 and in our plots we normalize the reward an agent received by the theoretical optimal mean return of $(1-\frac{1}{n})^n$ so that an optimal policy will receive an average episodic return of 1. This environment requires the agent's exploration method to be able to deal with stochasticity.

Cartpole Swingup Cartpole Swingup is a sparse reward control task with continuous states and discrete actions, similar to the classic Cartpole environment, except that the pole starts in a downward position and the agent has to explore to find that reward is received when the pole is upright. The agent also receives negative reward for moving the cart, disincentivizing exploration. This is a difficult exploration task in a continuous state space.

Mountain Car Mountain Car is a control task with continuous states and discrete actions, where the agent has to drive an under-powered car up a hill by building up momentum in a valley. In BSuite's implementation, an episode is 1000 steps and cancels early when the agent reaches the goal. The agent receives -1 reward at every time step, and has to learn to end the episode as fast as possible to maximize reward. Exploring to

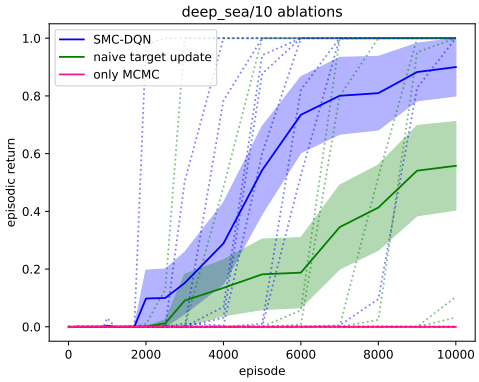

Figure 2.4: Learning curves on 30×30 deep sea for 2 ablations compared to SMC-DQN. The solid line is the mean of 10 seeds for 'naive target update' and 5 seeds for 'only MCMC'. Shaded areas denote the standard error of the mean and dashed lines show individual seeds.

find the goal state is not necessarily difficult in this environment, but finding an optimal policy that reaches the goal state quickly can be difficult.

2.5.2. BASELINES AND HYPER-PARAMETERS

Baselines We compare against the baseline Bootstrapped DQN agents in BSuite with and without randomized prior functions, since our algorithm can be considered an extension to the Bootstrapped DQN agent without priors, and the bootstrapped DQN agent with priors is also similar to our method and known to be a strong baseline in the exploration tasks that we consider.

Hyper-parameters We test all our agents with ensembles of size 10. For SMC-DQN, we use the same hyper-parameters across each experiment, except for Cartpole Swingup, where we set the standard deviation of the likelihood to $\sigma = 1$ instead of $\sigma = 0.1$ due to the much larger scale of cumulative reward in Cartpole Swingup. For our baselines, we use the hyper-parameters provided by BSuite. For the exact hyper-parameters we refer to the Supplementary Material.

2.5.3. EXPERIMENTAL RESULTS

Figure 2.3 shows the performance of the agents on each task. It can be seen that SMC-DQN outperforms BootDQN without priors on all our benchmarks. On Deep Sea it achieves comparable performance to BootDQN with priors, and significantly outperforms BootDQN with priors on Cartpole Swingup, where BootDQN at this ensemble size fails to learn a meaningful policy even with prior functions. When increasing the prior scale on BootDQN with priors we still did not observe effective exploration with an ensemble of size 10 on Cartpole Swingup. We refer to the supplementary material for these results. Further, on Mountain Car SMC-DQN learns at the same speed as BootDQN with priors in the beginning, but converges to a slightly better policy.

To confirm that SMC-DQN's uncertainty quantification is sensible, Figure 2.5 shows

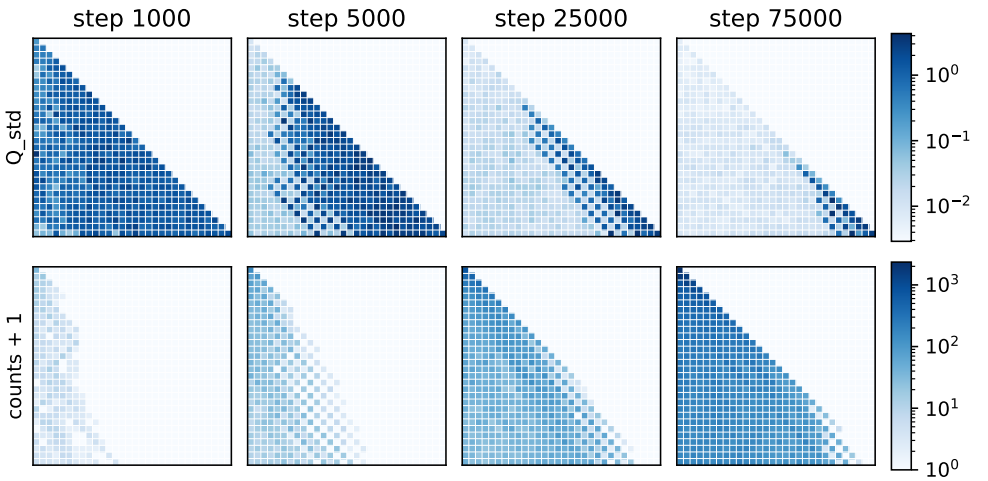

Figure 2.5: Graphical view of the standard deviation and visitation counts of the values during training on deep_sea/10. Each pixel shows standard deviation in the maximum Q-value (top row) of that state as predicted by the ensemble. The bottom row shows the logarithmic visitation counts. In each figure, the top left state is the initial state, and the bottom right state is the goal state. The diagonal is the only sequence of states that results in reward. States in the upper triangle are unreachable.

the standard deviation in Q-values as proxy to uncertainty, and visitation counts of SMC-DQN on 30×30 Deep Sea. Each square represents a state in Deep Sea, with the starting state in the top left and the bottom right state being the only state that results in positive reward. It is clearly visible how the uncertainty in values stays high for unvisited states, meaning that the approximated posterior distribution can correctly lead the agent towards under-explored areas.

2.5.4. ABLATION STUDY

Finally, in Figure 2.4 we study the effect of individual components of SMC-DQN by running two ablations.

Only MCMC First, to test whether SMC aids in approximating the posterior, we run an ablated agent which only uses the MCMC kernel, the Symplectic Euler Langevin scheme, to approximate the posterior.

The weights $w_1, \ldots, w_n$ are dropped, and the models are now treated as an ensemble of independent markov chains. The SMC step is replaced by 100 steps of the MCMC kernel, which is the number of steps that SMC would run this kernel for on the final target distribution. The split between the new batch and old buffer is removed, meaning that all transitions are immediately added to the replay buffer. The posterior density is approximated with samples from the replay buffer:

$$\log p(\theta_i|\boldsymbol{\theta}', \mathscr{D}) \approx \log p(\theta) + \sum_{i=1}^{B} -\frac{1}{2\sigma^2}\left(Q_\theta(s_i, a_i) - r_i -' \max_{a'} \gamma Q_{\theta'_i}(s', a')\right)^2,$$

for each $i = 1, \ldots, n$. All other hyperparameters are left unchanged.

This agent fails to observe any reward, suggesting that the MCMC kernel by itself can not keep a diverse set of models to explore with.

Naive target update Further, we check whether the new target update, which is theoretically required to maintain an approximation of the posterior, also improves reward in RL experiments. We run an ablated agent that naively updates its targets, omitting the SMC step.

The only change made to the algorithm is to remove the SMC step after updating the targets. This means that when the target networks are updated, we simply set $\theta'_i \leftarrow \theta_i$ for every $i = 1,\dots,n$, and leave the weights and parameters $\theta_1,\dots,\theta_n$ unchanged until the main update when the next batch of transitions is collected.

It can be observed that this agent can still receive reward, but on average later and less reliably.

2.5.5. DISCUSSION

Our results show a clear gap between Deep Sea and the continuous environments in the performance relative to the baseline. We hypothesize that this is due to the fact that the likelihood does not explain the one-hot encoded environment Deep Sea very well. On the continuous environments, agents can exploit the generalization capabilities of neural networks, allowing the posterior to model sensible generalization behaviours. However, since Deep Sea is under the hood a tabular environment, a perfect uncertainty mechanism on Deep Sea would model every state as independent unless they are connected, while the Bayesian posterior attempts to generalize over all states through the dependency on θ. This means that the posterior distribution does not necessarily provide more accurate uncertainty quantification than an ensemble with randomized priors of large scale.

2.6. RELATED WORK

Approximating the Bayesian posterior over Q-values to quantify uncertainty and drive exploration in Reinforcement Learning has been previously studied by several prior works. Monte Carlo Dropout [Gal and Ghahramani, 2016] is a Variational Inference method that uses dropout layers, which probabilistically disables neural network connections to create stochasticity in the outputs. The network together with the dropout probabilities are then trained so that the outputs match the predictive posterior. Furthermore, NoisyNets [Fortunato et al., 2019] adds stochasticity by modelling the posterior as independent normal distributions for each weight. Azizzadenesheli et al. [2018] replaces the last layer of a neural network with Bayesian Linear Regression, inferring the posterior distribution only over the parameters in the last layer. Epistemic Value Estimation [Schmitt et al., 2023] increases the size of the model and uses a Laplace approximation to approximate the posterior distribution over the full set of parameters. Due to the complex and non-linear nature of neural networks, it is not clear in these variational inference methods how the choice of model class for the posterior distribution affects the approximation accuracy. Our algorithm, on the other hand, uses MCMC methods to approximate the posterior, which is not restricted to a model class that has to be picked in advance. This

leads to unbiased approximations in theory, although due to practical considerations using an MCMC method to approximate a complex posterior distribution is not a simple task.

Langevin DQN [Dwaracherla and Roy, 2021] is an MCMC-based algorithm that approximates the posterior using Langevin Dynamics, which essentially perturbs the gradients of a DQN agent with normally distributed noise. Similarly, LMCDQN [Ishfaq et al., 2023] uses a more intricate MCMC kernel with a preconditioner to improve computational performance. These methods are comparable to the inner MCMC kernels inside the SMC algorithm in our agent, and could be used as drop-in replacements to the Symplectic Euler Langevin algorithm that we used for its empirically established accuracy [Wenzel et al., 2020]. While Langevin DQN and LMCDQN can also be used to train an ensemble of Q-networks, SMC-DQN differs in that SMC theoretically and practically ties together the ensemble members as opposed to running separate Markov Chains. Furthermore, we use a mixture distribution as likelihood so that the ensemble consistently models one posterior. Further, we update our target parameters in a theoretically sound manner that takes into account that the posterior distribution is conditioned on all target parameters jointly.

2.7. CONCLUSION

This paper introduced the novel idea of using sequential Monte Carlo to train an ensemble in order to approximate the Bayesian posterior distribution. Specifically, we modified the BootDQN algorithm to use Sequential Monte Carlo, thus keeping track of a posterior over the Q-values in a theoretically sound manner.

We found that such an approach is able to maintain a diverse set of models that can drive exploration in difficult-to-explore environments such as Deep Sea and Cartpole Swingup. Especially in continuous state environments, the uncertainty quantification provided by the posterior distribution leads to better exploration compared to our baselines.

In the future, we intend to investigate the influence of the choice of prior and likelihood, derive methods to synthesize meaningful priors and likelihoods, as well as extend our method to more intricate reinforcement-learning architectures.

Sequential Monte Carlo hyperparameters	
$\pi(\theta)$ (prior)	i.i.d. $\mathcal{N}(0,1)$
σ (likelihood std)	0.1
B (batch size)	32
MCMC steps (main)	10
MCMC steps (target)	10
Symplectic Euler Langevin Scheme hyperparameters	
ℓ (learning rate)	10^{-3}
cycle length	20
β	0.98
h (step size)	$\sqrt{\frac{\ell}{\mathtt{n_{data}}}}$
γ	$\frac{1}{h}$

Table 2.1: Hyperparameters for SMC in the supervised learning experiment.

2.A. SUPERVISED LEARNING EXPERIMENT

This experiment tests posterior approximation methods in an ideal setting. The data is exactly drawn from the assumed likelihood and prior. The neural network is a fully connected network with two hidden layers of size 100 and 10. The data is generated by sampling x i.i.d. from a uniform mixture of three normal distributions with means -1, 0, and 1 and standard deviation 0.25. The labels y are generated by randomly sampling a neural network $\theta_{\mathtt{prior}}$ from the prior $p(\theta)$ and setting $y_i = f_{\theta_{\mathtt{prior}}}(x_i) + \epsilon_i$, where $\epsilon_i \sim \mathcal{N}(0, 0.25)$.

The hyperparameters for SMC are shown in Table 2.1. For randomized prior functions, the prior function was drawn from the same prior as used for SMC and HMC.

2.B. REINFORCEMENT LEARNING EXPERIMENTS

For every agent, the Q-value network is a fully connected neural network with two layers of size 50. For Deep Sea, the observation is flattened before the first layer. The hidden units have leaky ReLU activations. Action selection is unchanged and done by picking one ensemble member uniformly at random, and acting greedily with respect to that network for the rest of the episode. Each ensemble member has access to all data collected by other ensemble members. No noise is added to the TD errors. We use an ensemble size of 10 for every agent.

The hyperparameters used for SMC-DQN are shown in Table 2.2, and the hyperparameters for BootDQN with and without priors are shown in Table 2.3.

The hyperparameters for BootDQN + prior and BootDQN are the default hyperparameters in BSuite. In addition to the default BSuite hyperparameters, Figure 2.6 also shows results for our baseline on Cartpole with prior scales 15 and 30 to incentivize more exploration, but we did not observe significantly better results than reported in the main text.

Sequential Monte Carlo hyper-parameters	
$\pi(\theta)$ (prior)	i.i.d. $\mathcal{N}(0,1)$
σ (likelihood std)	0.1 (1.0 in cartpole)
B (batch size)	128
MCMC steps (main)	100
MCMC steps (target)	100
Target ESS	10%
Reinforcement Learning hyper-parameters	
Buffer size	∞
Main update frequency	B
Target update frequency	B
Symplectic Euler Langevin Scheme hyper-parameters	
ℓ (learning rate)	10^{-3}
cycle length	20
β	0.98
h (step size)	$\sqrt{\frac{\ell}{\mathtt{n_{data}}}}$
γ	$\frac{1}{h}$

Table 2.2: Hyper-parameters of SMC-DQN in the RL experiments.

BootDQN Hyper-parameters	
Learning rate	0.001
Optimizer	Adam ($\beta_1 = 0.9, \beta_2 = 0.999$)
Update frequency	every step
Target update frequency	every 4 steps
batch size	32
prior scale (if +priors)	5

Table 2.3: Hyper-parameters of BootDQN(+prior) in the RL experiments.

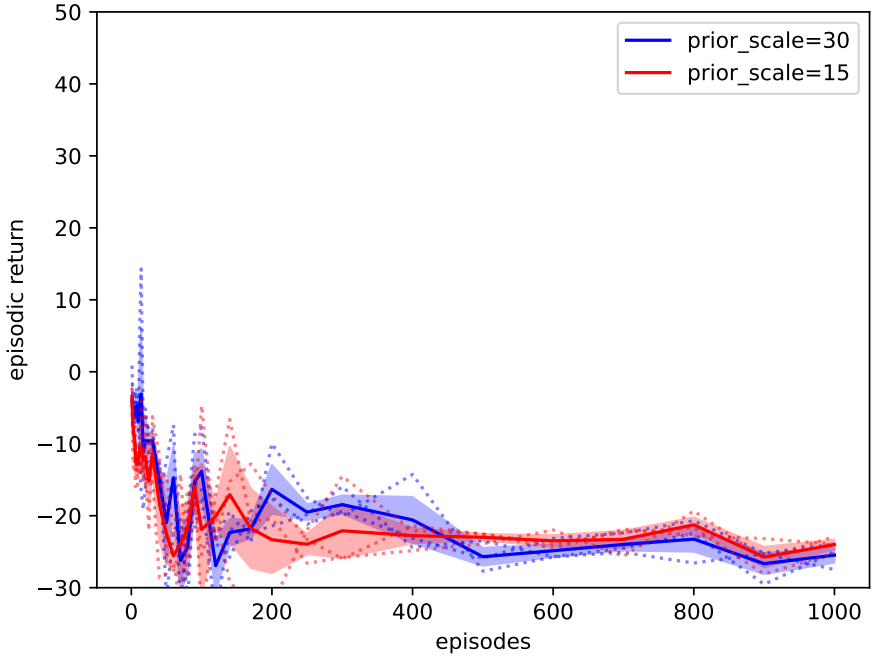

Figure 2.6: BootDQN with prior functions evaluated on Cartpole for several values of prior scale. The agent fails to find a performing policy.

3

EPISTEMIC BELLMAN OPERATORS

The previous chapter required making a design decision on how to treat the uncertainty in the value of the next state. This is an important decision which is made seemingly differently in other Bayesian reinforcement learning algorithms. This raises the question: Is there a theoretically correct decision? At the same time, it is unclear generally whether the loop of posterior inference and updating target values can be expected to converge. In this chapter, we answer both these questions.

In Section 3.3 we discuss the theoretical gap in interleaving posterior updates and target updates. In Section 3.3.2 we sketch a visual explanation that seemingly different algorithms actually approximate the same approach. In Section 3.4 we formalize this approach, and show that in tabular reinforcement learning settings this converges to a solution. Finally, in Section 3.5 we rediscover an existing algorithm from a different lens and adapt a widely adopted deep model-free reinforcement learning algorithm to an uncertainty aware variant making use of our theory.

This chapter is based on Pascal R. van der Vaart, Matthijs T. J. Spaan and Neil Yorke-Smith. "Epistemic Bellman Operators". In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2025

3.1. INTRODUCTION

Reinforcement learning (RL) algorithms have surpassed humans' ability in many games [Mnih et al., 2015, Schrittwieser et al., 2020], and have now also found success in real world problems such as controlling plasma in a nuclear fusion reactor [Degrave et al., 2022], video compression [Mandhane et al., 2022], large language models [Ouyang et al., 2022] and algorithm design [Fawzi et al., 2022, Mankowitz et al., 2023]. However, even for relatively simple tasks, algorithms still require many simulations or real interactions to learn a strong policy, making them inefficient. One approach to attack this problem is by making algorithms aware of their epistemic uncertainty, which is uncertainty caused by a lack of data. This allows them to explore only parts of the problem that are still uncertain, decreasing the total amount of interactions required.

However, proper uncertainty quantification is still an open problem in reinforcement learning. Many techniques from supervised learning, such as ensembles [Dietterich, 2000, Lakshminarayanan et al., 2017] and Bayesian methods [Chen et al., 2014, Liu and Wang, 2016, D'Angelo and Fortuin, 2021, Wenzel et al., 2020], have found success in practice when applied to supervised learning tasks with labelled data. However, in reinforcement learning data is not labelled with a ground truth, and instead the label for the current state is a self-supervised bootstrap from the label of the next state, known as the target value. Uncertainty quantification in RL must consider this sequential nature. At the heart of this problem is the fact that uncertainty in the current state should include the uncertainty in the target values, which is the uncertainty in the future states.

Adaptations of uncertainty quantification methods from supervised learning have been applied to reinforcement learning settings [Osband et al., 2016, 2018, Fortunato et al., 2019, Azizzadenesheli et al., 2018, Burda et al., 2018, Dwaracherla and Roy, 2021, Schmitt et al., 2023, Van der Vaart et al., 2024] with good practical results, but there is no guarantee that the way these algorithms treat the uncertainty in the successor state leads to a theoretically sound algorithm, in the sense that the uncertainty quantification aspect can be expected to converge to a solution at all. At least guaranteeing that these methods work in potentially simplified scenarios is essential for the adoption of uncertainty quantification in algorithms in the real world. Furthermore, some algorithms seemingly disagree in their decisions on how to treat the uncertainty in the target values.

When adapting Deep Q-learning (DQN)-style algorithms to uncertainty aware algorithms like BootDQN [Osband et al., 2016], EVE [Schmitt et al., 2023], Langevin-DQN [Dwaracherla and Roy, 2021], LMCDQN [Ishfaq et al., 2023], SMC-DQN [Van der Vaart et al., 2024] and BDQN [Azizzadenesheli et al., 2018], there are decisions to be made about how to use and update the target parameters. Generally, these algorithms condition their posterior on a posterior of the target parameters. As a main problem, we highlight that there is no guarantee that the process of repeatedly updating the current distribution, conditioned on the distribution over target parameters, and copying it to the target parameters will converge to a limiting distribution.

Recently, Fellows et al. [2021] studied this problem theoretically and contended that Bayesian model-free reinforcement learning algorithms create a posterior over Bellman operators. They showed that the posterior converges to the true Bellman operator in the limit of infinite data. We instead take an arguably more natural and direct approach, and show that the problem can be formulated as a generic Bellman operator that works on

distributions.

Specifically, our contributions are as follows:
1) We introduce Epistemic Bellman operators as a tool to analyze existing algorithms and develop theoretically sound uncertainty aware RL algorithms. Our unified framework formalizes the process of conditioning on distributions over target parameters.
2) We prove that Epistemic Bellman operators are contractions, implying that the process of interleaving posterior inference and target updates converges to a consistent fixed point for a general class of distributions and return estimators. Furthermore, we show that the mean of the fixed point of an Epistemic Bellman operator for policy evaluation is the fixed point of its non-epistemic counterpart.
3) We highlight the utility of Epistemic Bellman operators by analyzing an existing Bayesian Q-learning algorithm and alleviating an overestimation problem predicted by our theory. Furthermore, we develop a novel uncertainty aware version of Proximal Policy Optimization that clips less aggressively whenever it is certain about its advantages, and show improved performance in several environments.

3.2. BACKGROUND

3.2.1. MARKOV DECISION PROCESSES

We focus on Markov Decision Processes (MDP) with infinite horizon in the discounted reward setting. Formally, a Markov Decision Process is a tuple $(\mathcal{S}, \mathcal{A}, T, R, \gamma)$ of a state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $T : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$, reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ and discount factor $0 \leq \gamma < 1$. At each time step t, an agent observes the current state s_t, chooses an action $a_t \sim \pi(s_t)$ according to its policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$, and receives reward $r_t = R(s_t, a_t)$. The goal of reinforcement learning is to find a policy π that maximizes the discounted cumulative reward $\mathbb{E}_{T,\pi}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$. Of central importance is the Q-function

$$Q^\pi(s,a) = R(s,a) + \mathbb{E}_{T,\pi}\left[\sum_{t=1}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a\right],$$

denoting the expected discounted future reward if the agent executes action a in state s and then follows the policy π.

In a tabular setting, we represent the reward function, transition function and policy as vectors and matrices $R \in \mathbb{R}^{|S||A|}$, $T \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|\times|\mathcal{S}|}$, $\pi \in \mathscr{R}^{|S||A|}$. The Bellman operator for a policy π can then be written as

$$B^\pi_{T,R} Q = R + \gamma T^\pi Q,$$

where $T^\pi \in \mathbb{R}^{|S||A|\times|S||A|}$ is the transition function from state-action to state-action induced by the transition function T and the policy π, defined by

$$\begin{aligned}(T^\pi)_{sas'a'} &= \mathbb{P}(s_{t+1}, a_{t+1} = s', a' \mid s_t, a_t = s, a) \\ &= T_{sas'}\pi_{s'a'}.\end{aligned} \tag{3.1}$$

Since the transition function T and reward function R are assumed to be unknown to the agent, computing a strong policy requires exploration of the environment to learn which actions result in optimal return.

3.2.2. MODEL-FREE REINFORCEMENT LEARNING

Typically interesting problems have large states and action spaces, making it difficult to learn the transition and reward functions. Model-free algorithms such as actor-critics [Mnih et al., 2016, Schulman et al., 2017, Haarnoja et al., 2018] and Q-learning [Mnih et al., 2015] bypass this step and instead aim to learn a good policy or the values of a good policy directly, without estimating T and R.

A common component is to learn the values or Q-values by representing them by a neural network and minimizing the squared temporal difference loss on a dataset $\mathscr{D}$:

$$\begin{aligned} L_{TD}(\theta, \theta', \mathscr{D}) &= \sum_{(s,a,r,s') \in \mathscr{D}} TD(\theta, \theta', (s,a,r,s'))^2 \\ &= \sum_{(s,a,r,s') \in \mathscr{D}} \left[Q_\theta(s,a) - r - \gamma G(\theta', s') \right]^2, \end{aligned} \tag{3.2}$$

where $G(\theta', s')$ is some return estimator usually depending on a bootstrap from a target network θ' Mnih et al. [2015]. Examples are $G(\theta', s') = \max_{a'} Q_{\theta'}(s', a')$ in the case of one step Q-learning, or $G(\theta', a') = \sum_{a' \in A} \pi(a'|s') Q_{\theta'}(s', a')$ in the case of policy evaluation in actor-critics. The target network specifies what we believe the value is in the next state, and is updated periodically $\theta' \leftarrow \theta$ at a slower pace than θ to keep the target values $G(\theta', s')$ more stable for regression.

Agents use empirically observed transitions (s, a, r, s') to learn these models, requiring exploration to sufficiently cover the environment to achieve accurate values. Quantifying uncertainty in the value models can greatly improve the exploration capability of reinforcement learning algorithms through Thompson Sampling [Osband et al., 2016, 2018, O'Donoghue et al., 2018, Fortunato et al., 2019, Schmitt et al., 2023, Azizzadenesheli et al., 2018, Dwaracherla and Roy, 2021] or exploration bonuses [Ostrovski et al., 2017, Bellemare et al., 2016, Burda et al., 2018]. Furthermore, uncertainty quantification can also aid in general stability of algorithms by reweighting Bellman errors [Lee et al., 2021].

3.2.3. BAYESIAN VALUE LEARNING

One method to quantify uncertainty is through Bayesian algorithms. Generally, a Bayesian neural network is any neural network parameterized by $\theta \in \Theta$ where one attempts to model the posterior distribution

$$p(\theta|\mathscr{D}) = \frac{p(\mathscr{D}|\theta) p(\theta)}{\int p(\mathscr{D}|\theta) p(\theta) d(\theta)},$$

where $p(\mathscr{D}|\theta)$ is the likelihood, $p(\theta)$ is a prior and $\mathscr{D}$ is some data set. The posterior density $p(\theta|\mathscr{D})$ signifies how likely values of θ are, and is a natural method to model uncertainty as a distribution.

To equip an agent with uncertainty quantification, a posterior distribution over the parameters of a Q-function can be constructed $p(\theta|\mathscr{D}, \theta') \propto p(\mathscr{D}|\theta, \theta') p(\theta)$. Since the squared error loss is proportional to the log-density of a normal distribution, defining

$$p(\mathscr{D}|\theta, \theta') = \exp\left(- \sum_{(s,a,r,s') \in \mathscr{D}} \left[Q_\theta(s,a) - r - \gamma G(\theta', s') \right]^2 \right) \tag{3.3}$$

is a natural candidate for the likelihood when extending value learning algorithms to a Bayesian paradigm. This corresponds to the assumption that the temporal difference errors are normally distributed:

$$TD(\theta, \theta', (s, a, r, s')) \sim \mathcal{N}(0, \sigma^2). \tag{3.4}$$

While this assumption is in general not correct for every MDP, it is a convenient design choice and it should come as no surprise that several previous works have used this likelihood before [Osband et al., 2018, Schmitt et al., 2023, Dwaracherla and Roy, 2021, Azizzadenesheli et al., 2018, Ishfaq et al., 2023].

The likelihood $p(\mathcal{D}|\theta, \theta')$ and therefore also the posterior density $p(\theta|\mathcal{D}, \theta')$ does not only depend on the data, i.e., the observed transitions, it is also conditioned on the target values θ'. Handling this dependency is crucial for a theoretically sound algorithm that handles the sequential nature of uncertainty in this setting. Furthermore, posterior distributions are generally difficult to compute in practice, requiring approximate models. For example, BootDQN [Osband et al., 2016, 2018] uses ensembles, Langevin-DQN, LMCDQN and SMC-DQN [Dwaracherla and Roy, 2021, Ishfaq et al., 2023, Van der Vaart et al., 2024] use Monte Carlo methods, EVE [Schmitt et al., 2023] uses a Laplace approximation with diagonal covariance and BDQN [Azizzadenesheli et al., 2018] performs Bayesian inference over only the final layer of the Q-network.

3.3. PROBLEM STATEMENT

In this section we identify a key problem with model-free Bayesian reinforcement learning algorithms and motivate the value of our main contribution.

3.3.1. PROBLEMS WITH TARGET UPDATES

Roughly speaking, algorithms such as BootDQN, Langevin-DQN, LMCDQN, SMC-DQN, BDQN and EVE operate by interleaving steps

1. Infer a posterior given the current targets, $p_{\texttt{main}}(\theta|\mathcal{D}) = p(\theta|\mathcal{D}, \theta')$, where the targets are drawn or assumed to be from some distribution over targets $p_{\texttt{target}}(\theta')$.

2. Update the distribution over targets: $p_{\texttt{target}}(\theta) \leftarrow p_{\texttt{main}}(\theta|\mathcal{D}) = p(\theta|\mathcal{D}, \theta')$ to the current distribution over the main parameters θ.

This is analogous to the target update in many non-probabilistic algorithms that use temporal difference learning, and may seem like a reasonable adaptation to the Bayesian setting. However, for distributions there is no guarantee that this scheme converges, or is in fact well defined, since setting $p_{\texttt{target}}(\theta) \leftarrow p_{\texttt{main}}(\theta)$ is mathematically unsupported when $p_{\texttt{main}}(\theta)$ is a distribution that was conditioned on the target parameters. Furthermore, if this scheme does not converge to the same $p_{\texttt{main}}(\theta|\mathcal{D})$ for a fixed data set $\mathcal{D}$ and every starting distribution, it is not sensible to define a posterior $p_{\texttt{main}}(\theta|\mathcal{D})$ that is only conditioned on $\mathcal{D}$.

Fellows et al. [2021] propose interpreting the problem as inferring a posterior distribution over Bellman operators, and show convergence of the posterior to the true Bellman operator as more data is collected.

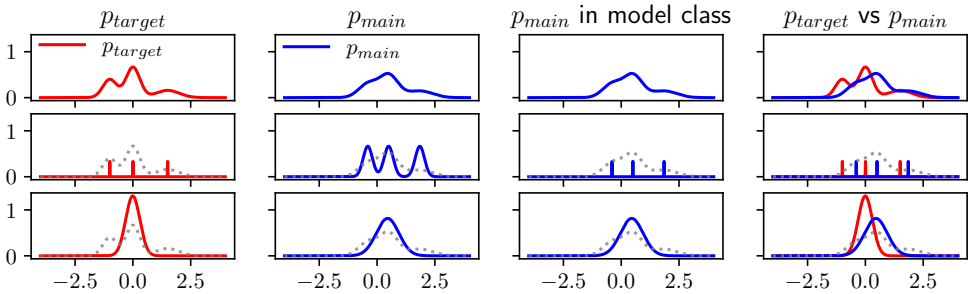

Figure 3.1: Plots of the distribution over the Q-value of a single state-action. The final column shows the difference between the target distribution (red) and the current distribution (blue). Rows are (1) idealized model class, (2) ensemble approximation (BootDQN), (3) Laplace approximation (EVE).

Instead, we propose a new Bellman operator that operates on distributions over Q-values, and prove that this operator is a contraction and has a fixed point. Roughly speaking, we show that an algorithm that alternates between updating a distribution conditioned on the targets, and updating the distribution over targets converges to a limiting distribution, proving that several common Bayesian algorithms which are special cases of our operator can be expected to converge, independent of the starting distribution.

3.3.2. VISUALIZING THE DISTRIBUTIONS

Before we introduce Epistemic Bellman Operators, we analyze which distributions Bayesian Q-learning algorithms actually attempt to approximate. To this end, we study BootDQN and EVE in a tabular setting, and assume there exists some idealized distribution over targets $Q' \sim p_{\texttt{target}}(Q)$ that our agent currently has. Furthermore, as in BootDQN and EVE, we are equipped with a likelihood

$$p(\mathscr{D}|Q,Q') \propto \exp\left(-\frac{1}{2\sigma^2}\sum_{(s,a,r,s')\in\mathscr{D}}\mathrm{TD}(Q,Q',s,a,r,s')^2\right),$$

also conditioned on a set of target values Q'.

This results in a posterior distribution

$$p(Q|\mathscr{D},Q') \propto p(\mathscr{D}|Q,Q')p(Q).$$

However, this distribution is conditioned on a single value for the targets and does not yet incorporate the fact that $Q' \sim p_{\texttt{target}}(Q)$, i.e., the uncertainty over the targets.

In the case of BootDQN, $p_{\texttt{target}}(Q)$ is modeled by the ensemble of target networks $\theta'_1,\ldots,\theta'_n$, and to approximate the posterior each ensemble member optimizes for its own loss $Q_i^* = \arg\max p(Q_i|\mathscr{D},Q'_i)$. On the other hand, EVE has a Laplace approximation for $p_{\texttt{target}}(Q)$, and updates the main distribution by sampling one $\tilde{Q}' \sim p_{\texttt{target}}(Q)$, maximizing $Q = \arg\max p(Q|\mathscr{D},\tilde{Q}')$ and also updating the Fisher information.

In our idealized setting, we can directly consider the marginalization of the condi-

tioned posterior over targets Q':

$$p_{\texttt{main}}(Q|\mathscr{D}) = \int p(Q|\mathscr{D}, q')\, d\, p_{\texttt{target}}(q').$$

Figure 3.1 shows a graphical presentation of this marginalization, together with BootDQN and EVE, in a simplified setting with an MDP with one state and one action. We assume the likelihood is a Gaussian on the one-step temporal difference:

$$\mathrm{TD}(Q, Q', s, a, r, s') = Q(s,a) - (r + \gamma Q'(s', a)),$$

where the policy is omitted for this one-action case. The top row is the idealized version of Bayesian model-free reinforcement learning algorithms. A distribution over the targets Q' defines a pushforward distribution over $BQ' = r + \gamma Q(s', a)$, which in turn implies a distribution over the main values which can exactly be inferred by a fully expressive model class. The second row contains a sketch of the situation with ensembles. The distribution $p_{\texttt{target}}$ is an ensemble, which together with the normal distribution likelihoods makes a mixture distribution over with modes $r + \gamma Q_i(s', a)$ for each ensemble member i. Estimating this distribution with an ensemble ideally returns an ensemble containing the modes of the new distribution, establishing the analogy between marginalization over Q' and simply applying the operator $BQ_i := r + \gamma Q_i(s', a)$ to each ensemble member individually, which is how BootDQN is trained.

For EVE, which uses Laplace approximations for its models, the target distribution is a normal distribution. The distribution for the current state is therefore also a normal distribution, and representing it in the model class of normal distributions returns a normal distribution.

Both BootDQN and EVE can be considered as approximations to this marginalization, approximating the integral with an ensemble in the case of BootDQN and a single sample from $p_{\texttt{target}}(q')$ in the case of EVE. After constructing an approximate $\tilde{p}_{\texttt{main}}(Q|\mathscr{D})$ each method then attempts to represent this distribution in their model class.

Considering this marginalization process, we can now define what it means for a well-defined posterior to exist. If the process of

$$p^{(k)}_{\texttt{main}}(Q|\mathscr{D}) = \int p(Q|\mathscr{D}, Q')\, d\, p^{(k)}_{\texttt{target}}(Q') \tag{3.5}$$

$$p^{(k+1)}_{\texttt{target}}(Q') = p^{(k)}_{\texttt{main}}(Q|\mathscr{D}) \tag{3.6}$$

$$k = k+1 \tag{3.7}$$

converges to the same limiting distribution $p(Q|\mathscr{D})^*$ for every starting $p^{(0)}_{\texttt{target}}(Q')$, the posterior distribution $p(Q|\mathscr{D})$ is well-defined. We formalize this process with the Epistemic Bellman Operator.

3.4. EPISTEMIC BELLMAN OPERATORS

For any Bellman operator or contraction $B_{\mathscr{D}}$, perhaps depending on some data set $\mathscr{D}$, we can define a pushforward distribution with additive noise as

$$p(Q|\mathscr{D}, Q') = \mathrm{Law}\left(B_{\mathscr{D}}(Q') + \epsilon_{\mathscr{D}}\right), \tag{3.8}$$

where Law(X) denotes the probability density of X. This is equivalent to the notion that the Q-values are distributed around the target values Q' with some *local* uncertainty $\epsilon_{\mathscr{D}}$, independent of Q'. This is a naturally occurring distribution in literature, since the posterior distribution of a normal likelihood with a normal prior takes this shape, which is commonly used in model-free deep RL literature [Osband et al., 2016, 2018, Schmitt et al., 2023, Fortunato et al., 2019, Azizzadenesheli et al., 2018, Dwaracherla and Roy, 2021, Ishfaq et al., 2023]. The Epistemic Bellman Operator for this distribution marginalizes the distribution over Q', and returns a new distribution.

Definition 1 (EBO)**.** For any measurable set A, let $\mathscr{P}(A)$ denote the set of probability distributions over A. Let $p(q|q')$ be a distribution over Q-values conditioned on target Q-values, e.g., Equation 3.8. We define the corresponding Epistemic Bellman Operator (EBO), as an operator $\mathscr{B}_p : \mathscr{P}(\mathbb{R}^{|S||A|}) \to \mathscr{P}(\mathbb{R}^{|S||A|})$, mapping distributions over Q-values to another distribution over Q-values by

$$\mathscr{B}_p P_Q(q) = \int p(q|q') dP_Q(q'). \tag{3.9}$$

When $p(q|q')$ is of the form Law$\left(B_{\mathscr{D}}(q') + \epsilon_{\mathscr{D}}\right)$, we can equivalently write Equation 3.9 as

$$\mathscr{B}_p P_Q = \text{Law}\left(B_{\mathscr{D}}(Q) + \epsilon_{\mathscr{D}}, Q \sim P_Q\right). \tag{3.10}$$

If the distribution $p(q|q') = \text{Law}(B_{\mathscr{D}}(Q) + \epsilon_{\mathscr{D}})$ has contracting properties, for example when $B_{\mathscr{D}}$ is a Bellman operator, it can be shown that the respective EBO is also a contraction. This is formalized in Theorem 1, whose proof is provided in Appendix 3.A.

Theorem 1 (Contraction)**.** *Let $\mathscr{Q} = (\mathbb{R}^{|S||A|}, \|.\|_\infty)$ be a metric space, $B_{\mathscr{D}}$ be a contraction on $\mathscr{Q}$, and let $p_B(q|q') = Law\left(B_{\mathscr{D}}(q') + \epsilon_{\mathscr{D}}\right)$ be a distribution over $\mathscr{Q}$ conditioned on target values in $\mathscr{Q}$.*

Then the corresponding Epistemic Bellman Operator $\mathscr{B}_p : \mathscr{P}(\mathscr{Q}) \to \mathscr{P}(\mathscr{Q})$ defined by Equation 3.10, where ϵ_D is independent of Q, is a W_ℓ-contraction on $\mathscr{P}(\mathscr{Q})$ for any $\ell \in [1, \infty)$.

This theorem implies that for any dataset $\mathscr{D}$, and any return estimator that is a contraction on Q-values, repeatedly applying an EBO to any starting distribution will converge to a fixed point. A consequence is that algorithms which interleave posterior inference with target distribution updates are theoretically sound in the sense that they converge to a unique solution $p(Q|\mathscr{D})$. Theorem 1 does not characterize the optimality of this solution, because this depends on the inner non-epistemic Bellman operator, which is typically the decisive factor for the functioning of an algorithm. For example, in the next section we will apply EBOs to the Optimal Bellman operator as well as Proximal Policy Optimization's return estimator, yielding two very different algorithms.

In the case of policy evaluation with a one-step Bellman Operator

$$BQ = R + \gamma T^\pi Q,$$

the fixed point of the EBO $\mathscr{B}$ is simple to characterize, and can be theoretically verified to be consistent with its non-epistemic counterpart. This can be extended to any affine B.

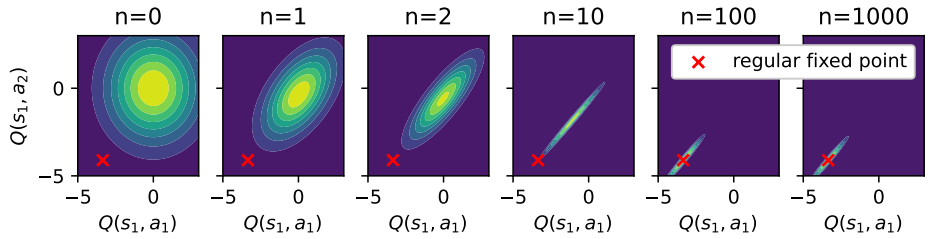

Figure 3.2: The Epistemic Bellman Operator applied iteratively to an initial distribution with a fixed policy in a single-state, two-actions MDP. The fixed point of the regular Bellman operator is in red.

Notably, the following theorem states that the mean of the fixed point is equal to the fixed point of the non-epistemic Bellman operator in $p(q|q')$ when it is affine and ϵ has mean zero. We refer to Appendix 3.A for the proof.

Theorem 2 (Mean of $\mathscr{B}$)**.** *Let $\mathscr{B}$ be the EBO corresponding to $p_B(q|q') = Law\big(B(q') + \epsilon\big)$ with $\mathbb{E}[\epsilon] = 0$. Let $P_B(Q)$ be the fixed point of $\mathscr{B}$, and Q_B be the fixed point of B. If B is an affine contraction, then $\mathbb{E}_{P_B}[Q] = Q_B$. Furthermore, writing $B(Q) = AQ + b$, the covariance $\Sigma_Q = \mathbb{E}_{P_B}\left[QQ^\top - Q_B Q_B^\top\right]$ is given by*

$$Vec(\Sigma_Q) = (I - A \otimes A)^{-1} Vec(\Sigma_\epsilon)$$

where $Vec(X)$ denotes the vectorization of X and $\otimes$ is the Kronecker product.

To showcase what our theorems state, we conduct an experiment in an MDP with one state and two actions so that the distributions are easy to visualize. We initialize a multivariate normal distribution, and iteratively apply the EBO. Figure 3.2 displays the density of the distribution over time, with the fixed point of the non-epistemic Bellman equation Q^π marked in red. It can be seen that the distributions converge to a normal distribution centered around Q^π, where the Q-values are strongly correlated. This correlation is expected, since both actions transition to the same state. Furthermore, Figure 6 in the appendix shows that the Wasserstein distance to the fixed point matches the theoretical contraction rate of γ.

3.5. USE CASES OF EPISTEMIC BELLMAN OPERATORS

In this section we highlight two main use cases for Epistemic Bellman Operators: gaining theoretical insight into existing methods by interpreting them with EBOs, and creating new methods using EBOs to guide the model updates.

3.5.1. THOMPSON SAMPLING WITH EBOS

Thompson sampling is a popular exploration algorithm [Azizzadenesheli et al., 2018, Dwaracherla and Roy, 2021, Ishfaq et al., 2023], making use of approximate sampling from a posterior distribution. More precisely, given a distribution P_Q of likely models, Thompson sampling samples a candidate model $Q \sim P_Q$ and acts greedily with respect to Q. We can model this behaviour with Epistemic Bellman Operators by taking the stan-

dard Optimal Bellman Operator as inner operator for our EBO:

$$p(q|q') = \text{Law}\left(R + \gamma T^{\pi^*_{q'}} q' + \epsilon\right),$$

where $\pi^*_{q'}$ denotes the greedy policy with respect to q'. The corresponding Epistemic Bellman Operator reads

$$\mathscr{B}P_Q = \text{Law}\left(R + \gamma T^{\pi^*_Q} Q + \epsilon_{\mathscr{D}}, Q \sim P_Q\right).$$

As a result of Theorem 1, it is known that this operator is a contraction and has a fixed point. However, since

$$\begin{aligned} &\mathbb{E}_{P_Q, P_\epsilon}[\max_a (Q(s,a) + \epsilon(s,a))] \\ &\geq \mathbb{E}_{P_\epsilon}[\max_a \mathbb{E}_{P_Q}[Q(s,a) + \epsilon(s,a)]] \\ &> \max_a \mathbb{E}_{P_Q}[Q(s,a)] \end{aligned} \tag{3.11}$$

when the event $\epsilon > 0$ has positive probability and $\mathbb{E}[\epsilon] \geq 0$, it can be predicted that the mean of the fixed point of $\mathscr{B}$ will overestimate the true values of the Thompson sampling policy, similar to the overestimation bias in Q-learning [Van Hasselt, 2010, Van Hasselt et al., 2016]. Epistemic Bellman Operators can remedy the overestimation in the same manner as in Q-learning, through sampling two independent samples from P_Q. This leads to the operator

$$\mathscr{B}_2 P_Q = \text{Law}\left(R + \gamma T^{\pi^*_{Q'}} Q + \epsilon_{\mathscr{D}}, Q, Q' \sim P_Q\right), \tag{3.12}$$

which reduces the estimation bias by selecting actions from an independent sample. We conjecture that this is operator is also a contraction under the same assumptions as Theorem 1, and we see in experiments that the values do converge.

Experiments To showcase this result, we run Thompson Sampling (TS) policies in a tabular environment, using Hamiltonian Monte Carlo, a standard MCMC algorithm, to approximately sample from the posterior, and using EBOs to directly sample from the exact distribution. Both methods are provided with unbiased estimators for T and R, so that any errors in the value models are purely due to bias in the algorithms. We then compare the mean of the sampled values to the true values achieved by the TS policy. The results are shown in Figure 3.3, with implementation details in Appendix 3.B. It can be seen that both the approximate sampler (MCMC) and exact sampler (EBO) overestimate the values with the same linear scaling in ϵ. However, using the double-sampling EBO, eliminates the bias. Furthermore, approximately sampling the fixed point of the double-sampling EBO with an MCMC algorithm also eliminates the bias. In agreement, Ishfaq et al. [2023] report that the double-Q sampling trick also helps MCMC methods in deep RL settings.

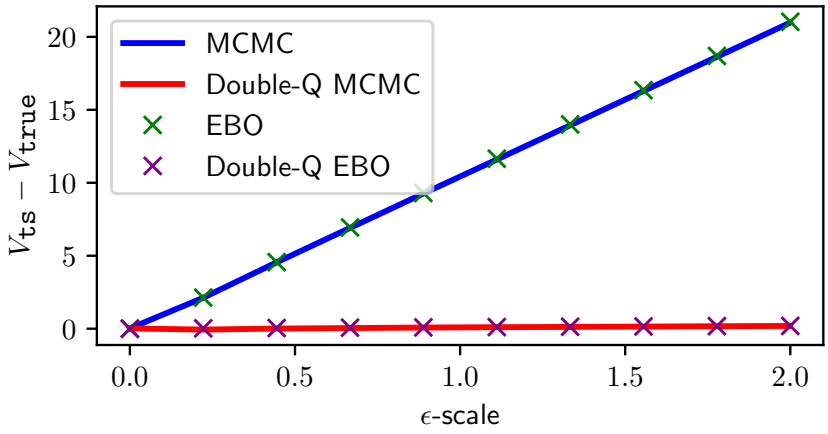

Figure 3.3: The gap between predicted values and true values of Thompson Sampling policies on a tabular MDP, with various local noise scales. Each line is the mean of 10 independent experiments.

3.5.2. EPISTEMIC CLIPPING PPO

Since Theorem 1 holds for any contraction, it is applicable to a wide range of return estimators used in practice. To showcase the generality of our results, we modify Proximal Policy Optimization (PPO) Schulman et al. [2017] into Epistemic Clipping PPO (ECPPO) by replacing the value models with a distributional model.

Proximal Policy Optimization PPO is a popular model-free reinforcement learning algorithm based on the actor-critic architecture. At a high level, it adapts actor critic architectures to a slightly off-policy setting so that multiple updates can be done on the same batch of data. It parametrizes a policy network π_θ and value network V. During rollouts of the MDP it follows the policy π_θ and estimates the advantage A_t of its policy with the following return estimator:

$$A_t = \delta_t + \gamma\lambda\delta_{t+1} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}, \tag{3.13}$$

$$\text{where } \delta_t = r_t + \gamma V(s_{t+1}) - V(s_t), \tag{3.14}$$

which is a mixture of temporal differences over several time steps where λ is a hyperparameter to decay future temporal differences. In the extreme cases $\lambda = 0$ or $\lambda = 1$, this entails taking the one step advantage $A_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ or Monte Carlo return $A_t = \sum_{j=t}^{T} \gamma^j r_j - V(s_t)$. The policy is then optimized to maximize the advantage through the following loss:

$$L^{PPO}(\theta) = \mathbb{E}[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-c, 1+c)A_t)], \tag{3.15}$$

where $r_t(\theta)$ denotes the policy ratio $\frac{\pi(\theta)(a_t|s_t)}{\pi_{\texttt{old}}(a_t|s_t)}$, $\pi_{\texttt{old}}$ is the rollout policy (i.e. the policy that originally collected the transition) and $\texttt{clip}(x, a, b)$ clips the value x to the range (a, b). This clipping procedure ensures that the policy π_θ does not stray too far from the policy that originally collected the data, as the mismatch between the two policies induces bias in the advantage estimation.

Adaptively Clipping An approximation of the posterior over $V(s)$ provides the agent with uncertainty quantification on the advantages, which we use to clip less aggressively in the policy loss whenever we are certain about the advantages. To this the original PPO loss is modified to

$$L^{ECPPO}(\theta) = \mathbb{E}[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1 - c\phi(U_t), 1 + c\phi(U_t))A_t)], \tag{3.16}$$

where U_t is an estimate of the uncertainty in A_t and ϕ is a monotonically decreasing function, such that the clipping range expands whenever U_t is low.

To approximate the distributions defined by the Epistemic Bellman Operator, we present two options: ensembles and a Laplace approximation. In the ensemble implementation of ECPPO, the value network of PPO is replaced by an ensemble $V_1(s), \dots, V_n(s)$, and the advantages are computed according to each ensemble member k independently:

$$A_t^{(k)} = \delta_t^{(k)} + \gamma\lambda\delta_{t+1}^{(k)} + \cdots + (\gamma\lambda)^{T-t+1}\delta_{T-1}^{(k)}, \tag{3.17}$$

$$\delta_t^{(k)} = r_t + \gamma V_k(s_{t+1}) - V_k(s_t). \tag{3.18}$$

As in standard PPO, the advantages are then normalized $\tilde{A}_t^{(k)} = \frac{A_t^{(k)} - \mu}{\sigma}$ using statistics μ, σ estimated from the minibatch, and the uncertainty is defined as

$$U_t = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(\tilde{A}_t^{(k)})^2 - (\frac{1}{n}\sum_{k=1}^{n}\tilde{A}_t^{(k)})^2},$$

which is the empirical standard deviation of the ensemble. The clipping range is modified by a function $\phi(U_t)$ such that $0.5 \leq \phi(U_t) \leq 2$. For exact specifications we refer to Appendix 3.B.

The Laplace-based version of ECPPO uses a Laplace approximation with diagonal covariance for the value network $V(s)$. To approximate the uncertainty U_t, it is important to keep in mind the covariance between the values within the same trajectory $V(s_t), V(s_{t+1}), \dots, V(s_T)$. To this end, the advantages A_t are computed with a set of candidate models $V_1(s), \dots V_n(s)$ drawn from the approximate posterior $\mathcal{N}(\theta^{MLE}, \frac{1}{n}\mathcal{I}(\theta^{MLE})^{-1})$, where θ^{MLE} is the MLE estimator and $\mathcal{I}(\theta^{MLE})$ is the Fisher Information. We refer to Daxberger et al. [2021] for a more in-depth overview of Laplace approximations for BNNs. The advantages and uncertainty are computed from $V_1(s), \dots V_n(s)$ analogously to the ensemble-based ECPPO. However, unlike the ensemble-based version, no gradients are computed for these candidate models as they are only used to compute targets. This makes the Laplace version more scalable.

Experiments We test the RL agent with base clipping hyperparameter of $c = 0.2$ on all discrete state environments in Gymnax [Lange, 2022], which includes environments from OpenAI Gym [Brockman et al., 2016], BSuite [Osband et al., 2020], MinAtar [Young and Tian, 2019], and several miscellaneous environments [Lange and Sprekeler, 2022, Miconi et al., 2018, Sutton et al., 1999, Wang et al., 2016], but excluding SimpleBandit-bsuite, which is non-sequential and trivial, and MemoryChain-bsuite, which is non-Markovian.

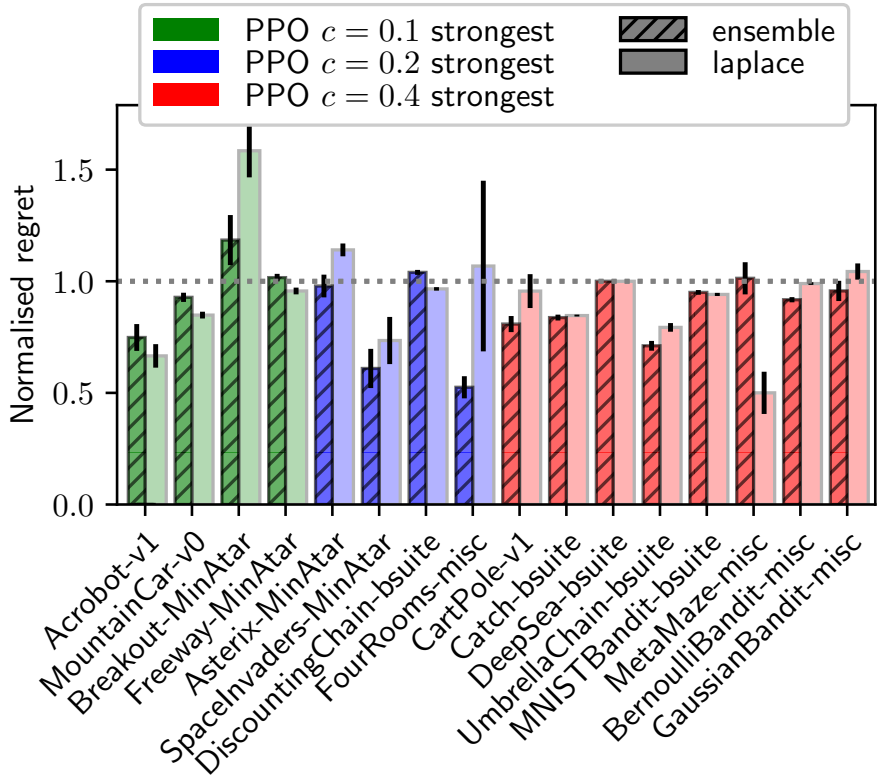

Figure 3.4: Regret of ECPPO with $c = 0.2$ relative to the regret of the baseline with $c = 0.2$ (lower is better). Environments are grouped and colour-coded by optimal baseline clipping parameter. Average of 20 seeds, with error bars denoting one standard error.

We compare results against the baseline version of PPO in PureJaxRL [Lu et al., 2022], which also has clipping ratio $c = 0.2$, and is tuned for these environments by the authors. Furthermore, since ECPPO is a modification to the clipping behaviour, we group environments by whether PPO improves with $c = 0.1$ and $c = 0.4$, which are the smallest and highest clipping ratio achievable by ECPPO.

Full experiment details and code are in Appendix 3.B, and all learning curves are in Appendix 3.C. To highlight how ECPPO improves over the baseline PPO with fixed c, Figure 3.4 shows the cumulative regret of ECPPO with $c = 0.2$ w.r.t. the strongest PPO baseline, normalised by the regret of the baseline with $c = 0.2$. The environments are grouped by whether decreasing or increasing c improves baseline performance. It is immediately visible that Ensemble-ECPPO dramatically improves performance across several environments, independent of whether high or low c is optimal in the specific environment, and without suffering major performance penalties in other environments. Laplace-ECPPO also improves performance in several independent on the optimal c, but becomes significantly worse on Breakout. Finally, we observe in Figure 8 (provided in Appendix 3.C) that the uncertainty quantification make sense in a qualitative manner

in the FourRooms environment, where uncertainty is high where the current policy has low support.

3.6. RELATED WORK

There is a large body of research for Bayesian methods in RL. On the practical side, there are algorithms such as BootDQN [Osband et al., 2016, 2018], EVE [Schmitt et al., 2023], BDQN [Azizzadenesheli et al., 2018], Langevin-DQN [Dwaracherla and Roy, 2021], LMCDQN Ishfaq et al. [2023] and SMC-DQN [Van der Vaart et al., 2024]. Our main theoretical result aims to theoretically ground these methods within a general framework by interpreting them as special cases of an EBO, which works on distributions, and prove that this is a contraction.

Operators that work on distributions are also a main focus in Distributional RL [Bellemare et al., 2017]. The goal in distributional RL is to model the distribution of returns, as opposed to learning only the mean. Distributional methods model the aleatoric uncertainty, which is the inherent randomness of returns due to the randomness in the policy and MDP. Instead, we focus on learning the mean of the returns, and compute a distribution over possible means given our observations to model the epistemic uncertainty on the mean. Furthermore, our operator naturally takes into account the dependency and covariance of the Q-values.

Dearden et al. [1998] discuss a similar operator, also providing convergence guarantees with a contraction argument. This result can be interpreted as a special case of our results with a specific return estimator and specific approximation class. Our main theorem instead applies to any return estimator with contractive properties.

Bayesian Bellman Operators [Fellows et al., 2021] also focus on the potentially problematic dependence on the target values when inferring posterior distributions over Q-functions. In their work, these problems are alleviated by interpreting Bayesian RL methods as inferring posterior distributions over Bellman Operators, while we directly consider distributions over Q-functions. Furthermore, they focus on a standard one-step Bellman operator with parameterized Q-functions, relying on gradient-based optimization theory to prove convergence in the limit of infinite data under assumptions on the data generating distributions. On the other hand, our results hold for any contraction operator and show existence and consistency for any data set.

3.7. CONCLUSION

We have introduced Epistemic Bellman Operators, which are operators that map a distribution over Q-values to the pushforward of regular Bellman operators with additive noise. We have shown that our operator generalizes several probabilistic reinforcement learning algorithms, unifying practical algorithms that appear to have dissimilar architectures. Furthermore, we have proven that Epistemic Bellman Operators are contractions, which implies that interleaving posterior inference and target updates converges to a fixed distribution and motivates these practical algorithms by showing consistency in tabular settings. We showed that the fixed point of an EBO is sensible when doing policy evaluation. Finally, we showcased the generality of our operators by studying an existing Bayesian Q-learning algorithm and modifying PPO into an uncertainty-aware

variant that outperforms the original algorithm in several environments.

In future research, the insights from our main theorem can aid in the design of new uncertainty-aware algorithms by guiding practical design choices toward theoretically sound approaches. Another research direction is to study more applications of uncertainty in reinforcement learning, other than exploration and the one presented here. Finally, we aim to investigate the influence of priors and likelihoods and study more suitable distributions than normal distributions.

3.A. PROOFS

Proof of Theorem 1. Finding a method to utilize the contraction properties of the inner Bellman operator is the main challenge in this proof. We will achieve this by defining an Epistemic Bellman operator on the joint space of two sets of Q-values with correlated noise. This will cause the noise to cancel later in the proof and allows us to use the inner Bellman operator as contraction.

Recall that the Wasserstein distance can be written as

$$W_p(P_X, P_Y) = \left(\inf_{R \in \mathscr{R}_{XY}} \mathbb{E}_{(X,Y)\sim R}(\|X - Y\|^p) \right)^{\frac{1}{p}},$$

where $\mathscr{R}_{XY}$ is the set of joint probability measures that has marginals P_X and P_Y, and $\|\cdot\|$ is the norm on $\mathbb{R}^{|S||A|}$ in which $B_{\mathscr{D}}$ is assumed to be a γ-contraction.

We can define an operator analogous to the EBO that works on the joint space instead:

$$\hat{\mathscr{B}}R(X,Y) = \text{Law}\left((B_{\mathscr{D}}(X) + \epsilon_{\mathscr{D}},\ B_{\mathscr{D}}(Y) + \epsilon_{\mathscr{D}}, (X,Y) \sim R(X,Y)\right),$$

where crucially both $\epsilon_{\mathscr{D}}$ are the **same** variable. We are allowed to choose $\epsilon_{\mathscr{D}}$ in this manner because it leaves the marginal distributions unchanged.

Because $\hat{\mathscr{B}}\bar{R}$ has marginals $\mathscr{B}P_X$ and $\mathscr{B}P_Y$ if $\bar{R}$ has marginals P_X and P_Y, we have

$$W_p(\mathscr{B}P_X, \mathscr{B}P_Y)^p = \inf_{R \in \mathscr{R}_{\mathscr{B}P_X, \mathscr{B}P_Y}} \mathbb{E}_R\left[\|X - Y\|^p\right] \leq \inf_{\bar{R} \in \mathscr{R}_{P_X, P_Y}} \mathbb{E}_{\hat{\mathscr{B}}\bar{R}}\left[\|X - Y\|^p\right]. \tag{3.19}$$

The inequality holds because the first infimum is over all R with marginals $\mathscr{B}P_X$ and $\mathscr{B}P_Y$, while the second infimum is restricted to the image of $\hat{\mathscr{B}}$.

Finally, we can conclude that

$$W_p(\mathscr{B}P_X, \mathscr{B}P_Y)^p \leq \inf_{\bar{R} \in \mathscr{R}_{P_X, P_Y}} \mathbb{E}_{\hat{\mathscr{B}}\bar{R}}\left[\|X - Y\|^p\right] = \inf_{\bar{R}} E_{\bar{R}} \|B_{\mathscr{D}}(X) + \epsilon_{\mathscr{D}} - B_{\mathscr{D}}(Y) - \epsilon_{\mathscr{D}}\|^p \tag{3.20}$$

$$= \inf_{\bar{R}} E_{\bar{R}} \|B_{\mathscr{D}}(X) - B_{\mathscr{D}}(Y)\|^p \leq \inf_{\bar{R}} E_{\bar{R}} \gamma^p \|X - Y\|^p \leq \gamma^p \inf_{\bar{R}} E_{\bar{R}} \|X - Y\|^p \tag{3.21}$$

$$= (\gamma W_p(P_X, P_Y))^p, \tag{3.22}$$

where in Equation 3.21 we use the fact that $B_{\mathscr{D}}$ is a γ-contraction in $\|.\|$. □

Proof of Theorem 2. Most of this proof relies on the fact that we can switch the order of expectations and affine functions.

Since P_B is a fixed point of $\mathscr{B}$, we have

$$\mathbb{E}_{P_B}[Q] = \mathbb{E}_{\mathscr{B}P_B}[Q] = \mathbb{E}_{P_\epsilon}\mathbb{E}_{P_B}[B(Q) + \epsilon] = \mathbb{E}_{P_B}[B(Q)] = B[\mathbb{E}_{P_B}[Q]], \tag{3.23}$$

where the second equality is the definition of the EBO, the third equality follows from independence of ϵ and Q, and the last equality uses the fact that B is an affine function, proving that $\mathbb{E}_{P_B}[Q]$ is a fixed point of B.

For the rest of the proof, we will write BQ as shorthand for $B(Q)$, and since B is affine we define A and b such that $BQ = AQ + b$.

For the covariance we have

$$\begin{aligned}
\mathbb{E}_{P_B}\left[QQ^\top\right] &= \mathbb{E}_{P_\epsilon}\left[\mathbb{E}_{P_B}\left[(BQ+\epsilon)(BQ+\epsilon)^\top\right]\right] \\
&= \mathbb{E}_{P_\epsilon}\left[\mathbb{E}_{P_B}\left[(BQ)(BQ)^\top + BQ\epsilon^\top + \epsilon(BQ)^\top + \epsilon\epsilon^\top\right]\right] \\
&= \mathbb{E}_{P_B}\left[(BQ)(BQ)^\top\right] + \mathbb{E}_{P_\epsilon}\mathbb{E}_{P_B}\left[BQ\epsilon^\top\right] + \mathbb{E}_{P_\epsilon}\mathbb{E}_{P_B}\left[\epsilon(BQ)^\top\right] + \mathbb{E}_{P_\epsilon}\left[\epsilon\epsilon^\top\right] \\
&= \mathbb{E}_{P_B}\left[(BQ)(BQ)^\top\right] + \mathbb{E}_{P_\epsilon}\left[\epsilon^\top\right]\mathbb{E}_{P_B}\left[BQ\right] + \mathbb{E}_{P_\epsilon}\left[\epsilon\right]\mathbb{E}_{P_B}\left[(BQ)^\top\right] + \mathbb{E}_{P_\epsilon}\left[\epsilon\epsilon^\top\right] \\
&= \mathbb{E}_{P_B}\left[(BQ)(BQ)^\top\right] + \mathbb{E}_{P_\epsilon}\left[\epsilon\epsilon^\top\right] \\
&= \mathbb{E}_{P_B}\left[(BQ)(BQ)^\top\right] + \Sigma_\epsilon \\
&= \mathbb{E}_{P_B}\left[(AQ+b)(Q^\top A^\top + b^\top\right] + \Sigma_\epsilon \\
&= A\mathbb{E}_{P_B}\left[QQ^\top\right]A^\top + b\mathbb{E}_{P_B}\left[Q^\top\right] + \mathbb{E}_{P_B}\left[Q\right]b^\top + bb^\top + \Sigma_\epsilon \\
&= A\mathbb{E}_{P_B}\left[QQ^\top\right]A^\top + b\mathbb{E}_{P_B}\left[Q^\top\right] + \mathbb{E}_{P_B}\left[Q\right]b^\top + bb^\top + \Sigma_\epsilon \\
&= A\mathbb{E}_{P_B}\left[QQ^\top\right]A^\top + bQ_B^\top + Q_Bb^\top + bb^\top + \Sigma_\epsilon,
\end{aligned} \tag{3.24}$$

and

$$Q_BQ_B^\top = (BQ_B)(BQ_B)^\top = AQ_BQ_B^\top A^\top + bQ_B^\top + Q_Bb^\top + bb^\top,$$

so

$$\begin{aligned}
\mathbb{E}_{P_B}\left[QQ^\top - Q_BQ_B^\top\right] &= A\mathbb{E}_{P_B}\left[QQ^\top\right]A^\top - AQ_BQ_B^\top A^\top + \Sigma_\epsilon \\
&= A\mathbb{E}_{P_B}\left[QQ^\top - Q_BQ_B^\top\right]A^\top + \Sigma_\epsilon.
\end{aligned} \tag{3.25}$$

Vectorizing both sides yields

$$\mathrm{Vec}(\mathbb{E}_{P_B}\left[QQ^\top - Q_BQ_B^\top\right]) - \mathrm{Vec}(A\mathbb{E}_{P_B}\left[QQ^\top - Q_BQ_B^\top\right]A^\top) = \tag{3.26}$$

$$\mathrm{Vec}(\mathbb{E}_{P_B}\left[QQ^\top - Q_BQ_B^\top\right]) - (A\otimes A)\mathrm{Vec}(\mathbb{E}_{P_B}\left[QQ^\top - Q_BQ_B^\top\right]) = \tag{3.27}$$

$$\mathrm{Vec}(\Sigma_\epsilon). \tag{3.28}$$

Finally, since B is a contraction, the absolute eigenvalues of A must be strictly smaller than 1. By basic properties of the Kronecker product, $A\otimes A$ then also has absolute eigenvalues strictly smaller than one. Therefore, $I-(A\otimes A)$ has only non-zero eigenvalues, and hence is invertible.

We conclude that

$$\mathrm{Vec}(\mathbb{E}_{P_B}\left[QQ^\top - Q_BQ_B^\top\right]) = (I - A\otimes A)^{-1}\mathrm{Vec}(\Sigma_\epsilon)$$

to finish the proof. □

3.B. EXPERIMENT DETAILS

3.B.1. EPISTEMIC BELLMAN OPERATOR

The MDP in this experiment has reward function $R = [0.05192758, -0.7084503]$, and the evaluated policy is $\pi = [0.4352794, 0.5647206]$. We use a standard 1 step Bellman operator B such that

$$BQ = R + \gamma T^\pi Q,$$

Algorithm 4: Pushforward sampling of EBO with i.i.d. ϵ

Input: Noise-scale σ
Output: A sample from the fixed point

Initialization:
$Q \leftarrow \mathbf{0}$
while not converged **do**
  $\epsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2)$
  $Q \leftarrow R + \gamma T^{\pi^*_Q} + \epsilon$
end while
Return Q

and the likelihood is given by $p(Q|q') = Bq' + \epsilon$ where $\epsilon \sim \mathcal{N}(0, \texttt{diag}(0.2, 0.2))$. The initial distribution is a normal distribution with mean 0 and covariance $\texttt{diag}(2,2)$.

Code that generated the figures is available at `github.com/pascal314/epistemic-bellman-operators`. These experiments were completed on a single CPU within a few minutes.

3.B.2. TABULAR THOMPSON SAMPLING

The MDP in this experiment has 30 states and 5 actions, with $\gamma = 0.9$. The transition function is drawn form a uniform dirichlet distribution, and the reward function is sampled i.i.d. from a normal distribution with standard deviation 0.1. We provide the true transition and reward function to all algorithms, but inject normally distributed noise ϵ with varying standard deviation.

Our MCMC sampler is a basic Hamiltonian Monte Carlo (HMC) sampler applied to the likelihood

$$p(Q) \propto \exp(\sum_{s,a} \frac{1}{2\sigma^2}(Q - R - \gamma T^{\pi^*_Q} Q))^2,$$

where σ is the true standard deviation of the injected noise. The Double-Q MCMC sampler is applied to the likelihood

$$p(Q, Q') \propto \exp\left(\sum_{s,a} \frac{1}{2\sigma^2}(Q - R - \gamma T^{\pi^*_{Q'}} Q)^2 + \frac{1}{2\sigma^2}(Q' - R - \gamma T^{\pi^*_Q} Q')^2\right),$$

The HMC hyperparameters are hand-tuned to have an acceptance rate around 0.9.

Our EBO sampler samples from the fixed point of the EBO by initializing a table of Q-values and repeatedly applying the EBO. The Double-Q EBO sampler uses the same approach but has two Q-tables and applies the Double-Q EBO instead. Pseudo-code for both methods is available in Algorithms 4 and 5.

Code is available at `github.com/pascal314/epistemic-bellman-operators` and runs in less than a minute.

Algorithm 5: Pushforward sampling of Double-Q EBO with i.i.d. ϵ

Input: Noise-scale σ
Output: A sample from the fixed point

Initialization:
$Q_1 \leftarrow \mathbf{0}$
$Q_2 \leftarrow \mathbf{0}$
while not converged **do**
  $\epsilon_1 \sim \mathcal{N}(\mathbf{0}, \sigma^2)$
  $\epsilon_2 \sim \mathcal{N}(\mathbf{0}, \sigma^2)$
  $Q_1 \leftarrow R + \gamma T^{\pi^*_{Q_2}} + \epsilon$
  $Q_2 \leftarrow R + \gamma T^{\pi^*_{Q_1}} + \epsilon$
end while
Return Q_1.

3.B.3. EPISTEMIC CLIPPING PPO (ECPPO)

Code for both Ensemble-ECPPO and Laplace-ECPPO is available at `github.com/pascal314/epistemic-bellman-operators`. We utilize PureJaxRL's implementation of PPO as baseline and backbone of our ECPPO implementations. Pseudo-code for the modified policy update is in Algorithm 7. Acting in the environment and the (clipped) value updates are unchanged from the baseline PPO.

For **Ensemble-ECCPPO**, we replace the value network with an ensemble of 5 value networks with the same architecture. The value update is applied to each ensemble member independently.

For **Laplace-ECPPO**, the value network is unchanged, but now also accompanied by a diagonal approximation of the inverse Fisher information matrix F. Whenever the value model is updated, the inverse Fisher approximation is also updated.

$$F \leftarrow F + \alpha\left(\frac{1}{\Lambda^2} + n \cdot \frac{g \odot g}{\sigma^2}\right), \tag{3.29}$$

where Λ and σ are hyperparameters representing the standard deviation of the prior and likelihood respectively, n is the total number of observations, and $g \odot g$ is the pointwise squared gradients of the value model loss with respect to the previous rollout. Pseudo-code to sample models from a Laplace approximation is provided in Algorithm 6. For a more detailed exposition of the workings of Laplace approximations, we refer to the references cited in the main paper.

ADAPTIVE CLIPPING

For each time step t the uncertainty U_t is computed by taking the standard deviation of the empirically observed advantages, as described in the main text. The clipping parameter of the policy ratio is then modified to $c\phi(U_t)$, where

$$\phi(u) = \frac{1}{2} + \frac{3}{2}\sigma\left(15 \cdot (-u + 0.3)\right),$$

Algorithm 6: Sampling from a Laplace Approximation

1: **Input:** Value network parameters θ, diagonal Fisher information matrix F at θ, and number of samples K.
2: **Output:** Samples $\{\theta_k\}_{k=1}^K$ from the Laplace approximation.
3: **for** each sample $k = 1$ to K **do**
4: Sample a vector $z_k \sim \mathcal{N}(0,1)$ with i.i.d. entries, of shape θ.
5: Scale z_k pointwise by the diagonal Fisher information

$$z_k \leftarrow z_k / \sqrt{F}$$

6: Shift z_k by the mean (i.e. the main parameters)

$$\theta_k = z_k + \theta$$

7: **end for**
8: **Return** the set of samples $\{\theta_k\}_{k=1}^K$.

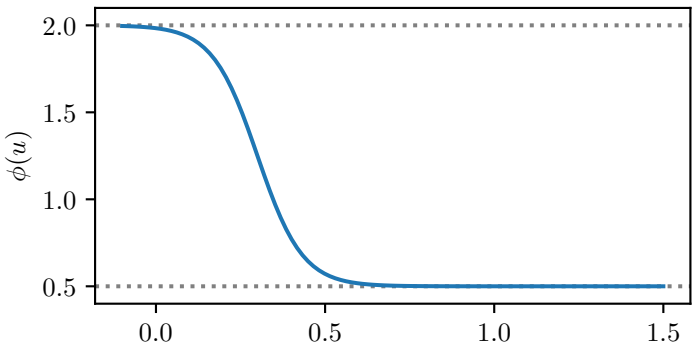

Figure 3.5: A plot of the function $\phi(u)$ used in our ECPPO experiments.

where σ is the sigmoid function.

We hand-picked this function as a simple candidate that maps $[0,\infty) \to [0.5, 2.0]$, so that ECPPO can either halve or double the clipping range based on the uncertainty. To find reasonable values for scaling and shifting the uncertainty, we conducted an initial experiment on Acrobot-v1 where we empirically evaluated the expected modification to the clipping $\mathbb{E}_U[\phi(U)]$. We then picked the values 15 and 0.3 so that $\mathbb{E}_U[\phi(U)] \approx 1$ on this environment. We did not conduct any other hyperparameter optimization based on algorithm performance to obtain $\phi(u)$. While we used a specifically shaped function, we note that any positive and monotonically decreasing function ϕ is a valid choice.

HYPERPARAMETERS

We left all further hyperparameters unmodified from the baseline. For completion, we list the hyperparameters of PureJaxRL's implementation in Table 3.1.

Algorithm 7: Epistemic Clipping PPO Policy update

1: **Input:** Initial policy parameters θ, batch of trajectories with k independently estimated advantages $(s_t, a_t, r_t, A_t^{(k)})_{t\geq 1}$. The $k-$th advantage is estimated by the $k-$th ensemble member in Ensemble-ECPPO, or the k-th candidate model in Laplace-ECPPO.
2: **Output:** Optimized policy parameters θ
3: Normalize advantages:

$$A_t^{(k)} \leftarrow \frac{A_t^{(k)} - \mu}{s},$$

with μ, s estimated from the full batch across all ensemble members combined.
4: Compute the probability ratio $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$.
5: Compute the uncertainties : $U_t = \sqrt{\frac{1}{n}\sum_{k=1}^{n}(A_t^{(k)})^2 - (\frac{1}{n}\sum_{k=1}^{n} A_t^{(k)})^2}$ {Empirical standard deviation}
6: Compute average advantage $A_t = \sum_{k=1}^{K} A_t^{(k)}$
7: Compute the objective:

$$L^{\text{ECPPO}}(\theta) = \mathbb{E}\left[\min\left(r_t(\theta)A_t, \text{clip}\left(r_t(\theta), 1 - c\phi(U_t), 1 + c\phi(U_t)\right)A_t\right)\right]$$

8: Perform gradient ascent on $L^{\text{ECPPO}}(\theta)$ using Adam
9: **Return** the optimized policy parameters θ.

Parallel environments	64
Rollout length	128
Epochs	4
Batch size	1024
γ	0.99
Λ	0.95
c	[0.1, 0.2, 0.4]
Entropy loss factor	0.01
Value loss factor	0.5
Max gradient norm	0.5

Table 3.1: Hyperparameters of the PPO Baseline

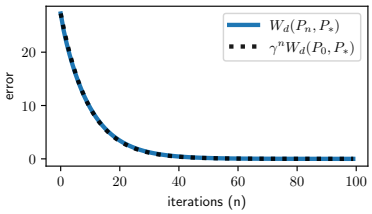

Figure 3.6: The Wasserstein Distance between distributions over Q-values and the fixed point of the EBO when iteratively applying the EBO (blue). The rate of contraction matches the predicted contraction rate γ (black, dashed)

The learning rate starts at 0.005 and follows a linear schedule down to 0 at the final episode.

The network architectures for both the actor and the value is a fully connected network with hidden sizes 64 and 64, and relu activations. The actor and value networks share no parameters. Epistemic Clipping PPO makes no modification to the actor network, and only replaces the value network with a distributional model.

For Ensemble-ECPPO, this is an ensemble with randomized priors of equivalent architecture, with outputs scaled by a factor of $\beta = 1$, which was not tuned.

For Laplace-ECPPO, this is a Laplace approximation. To find functional hyperparameters, We conducted a small hyperparameter search over $\Lambda = [0.1, 1, 2, 10]$, Fisher learning rate $\alpha = [10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}]$ and $\sigma = [0.1, 1, 2, 10]$ based only on performance of FourRooms. This resulted in prior scale $\lambda = 2.0$, fisher learning rate 10^{-2}, and likelihood $\sigma = 0.1$.

Each baseline and variant of ECPPO ran for 20 seeds on each environment. All experiments were completed on a single GPU, taking around one minute per agent per environment for all 20 seeds in parallel.

The regret with respect to the baseline is computed by first identifying the baseline agent that achieves the highest final performance. Then we compute for both ECPPO and the baseline with $c = 0.2$ the regret $R_{\texttt{agent}} = G^* - G_{\texttt{agent}}$ where G^* is the average cumulative reward over the training history (i.e. the area under the curve) for the best baseline, and $G_{\texttt{agent}}$ is the average cumulative reward for `agent`. Finally we report $\frac{G_{\texttt{ECPPO}}}{G_{c=0.2\texttt{baseline}}}$ in a barplot. To verify that ECPPO truly adapts the clipping rate and does not just consistently increase or decrease it, we color code the environments by which baseline performed best, to highlight that ECPPO can perform well independent of which c was optimal for the baseline PPO algorithm. Code that generated this figure is also available in the supplementary material.

3.C. ADDITIONAL FIGURES

Figure 3.7 shows the learning curves of ECPPO on each environment. It can be seen that in Pong, Acrobot, and FourRooms, Ensemble-ECPPO outperforms all baselines. Furthermore, on MountainCar, Freeway, Breakout and Space Invaders, lower c is optimal in the baselines, and Ensemble-ECPPO matches the best baseline. In Catch, UmbrellaChain, BernoulliBandit, and Gaussianbandit and Cartpole, higher c lets PPO learn faster, and

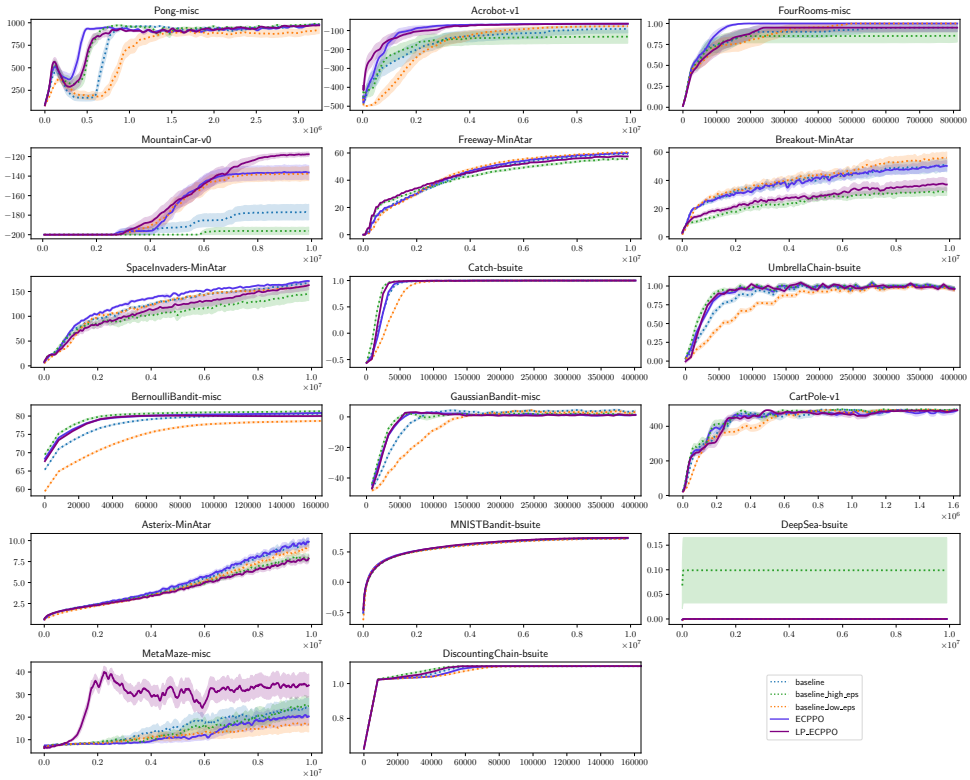

Figure 3.7: Mean learning curves of ECPPO and baseline PPO algorithms on 17 environments from Gymnax. Shaded areas denote the standard error of the mean, based on 20 seeds per agent per environment.

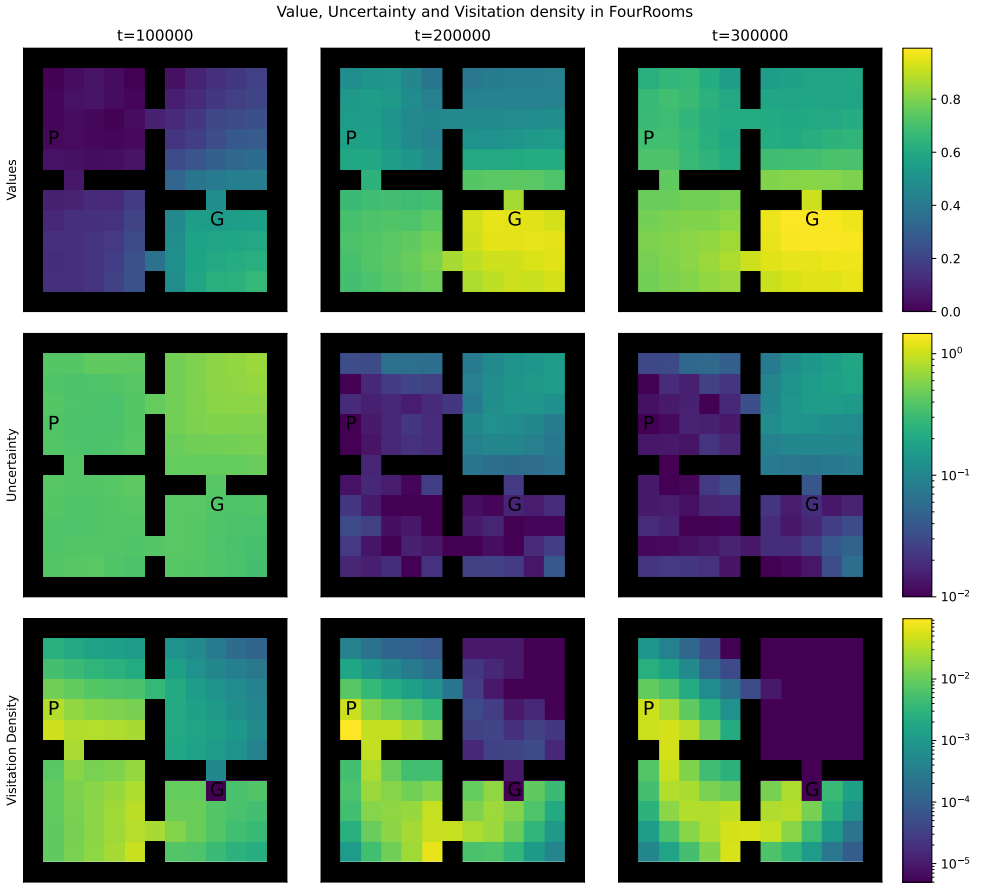

Figure 3.8: Value, uncertainty and state visitation density on FourRooms of ECPPO at several points during training. The starting position and goal position are denoted by *P* and *G* respectively.

Ensemble-ECPPO again matches the best baseline. In the rest of the environments the relationship between c and the performance of the baselines is not immediately clear, but Ensemble-ECPPO still matches or outperforms the strongest baseline, Except for DiscountingChain, where Ensemble-ECPPO is slightly behind the baseline with the respective c. Laplace-ECPPO also matches or exceeds the baseline in most environments, however, it has a clear disadvantage to the Ensemble-ECPPO and the baselines on Breakout and Asterix. We note that none of the agents solve 15×15 DeepSea reliably, which is unsurprising given that we do not explore actively.

4

PRIORS MATTER: ADDRESSING MISSPECIFICATION IN DEEP Q-LEARNING

In Chapter 2 and the experiments of Chapter 3 we made the assumption that temporal difference errors follow a Gaussian likelihood, and that the parameters of neural networks follow a Gaussian prior. This is a very common assumption in literature, but without any theoretical grounding. In this chapter we test common assumptions made in Bayesian reinforcement learning algorithms and propose a more suitable prior.

In Section 4.3 we show experimentally that there is a Cold Posterior Effect, where Bayesian algorithms perform better when their posterior temperature is tuned down, suggesting that there are issues with the likelihood and prior assumptions. In Section 4.4 we show that Gaussian priors are indeed not a suitable fit, and propose two new priors, one of which is trivial to implement and the other which is meta-learned on a pre-collected data set of trained agents. In Section 4.5 we statistically test the Gaussian likelihood assumption and show that likelihoods are also misspecified. We also discuss how choosing a better likelihood is more difficult than improving the prior. Finally, in Section 4.6 we experimentally show that our new priors outperform Gaussian likelihoods in terms of cumulative reward and reduce the Cold Posterior Effect.

This chapter is based on Pascal R. van der Vaart, Neil Yorke-Smith, and Matthijs T. J. Spaan. "Priors Matter: Addressing Misspecification in Deep Q-Learning", which is currently under review at the International Conference of Machine Learning, 2026.

4.1. INTRODUCTION

Reinforcement learning (RL) algorithms have many potential applications, but the exploration–exploitation trade off remains an open problem. Especially when real or simulated experiences are expensive, it is essential that RL agents can efficiently explore the environment to increase sample efficiency. Many exploration methods rely on the quantification of uncertainty, by assigning novelty bonuses [Ostrovski et al., 2017, Bellemare et al., 2016, Burda et al., 2018] or through sampling approaches such as Thompson sampling [Osband et al., 2016, 2018, O'Donoghue et al., 2018, Fortunato et al., 2019, Schmitt et al., 2023, Azizzadenesheli et al., 2018, Dwaracherla and Roy, 2021]. However, quantification of uncertainty for deep RL remains a challenging problem.

One uncertainty quantification method is through *Bayesian inference*, where an agent learns how likely certain models or values are, given a prior and the data it has observed. In theory, Bayesian algorithms have strong theoretical guarantees in well-defined settings. A well-known result in statistical learning theory is that Bayesian algorithms achieve optimal average loss with the correct prior and likelihood [Komaki, 1996]. Further, in bandit settings and the model-based RL, Thompson sampling is proven to have strong regret bounds [Agrawal and Goyal, 2012, Osband et al., 2013].

However, in the benchmarks currently used in deep reinforcement learning, the performance of Bayesian approaches depends heavily on the environment [Ishfaq et al., 2023, Van der Vaart et al., 2024], and they are sometimes outclassed by simple ensembles of maximum likelihood estimators, such as BootDQN [Osband et al., 2016, 2018]. The difference between practice and theory in reinforcement learning could be due to a variety of challenges that deep reinforcement learning imposes. This discrepancy in performance is not unique to reinforcement learning, however.

In deep supervised learning, Wenzel et al. [2020] identified that there is a *cold posterior effect*, where performance increases as the temperature of the target posterior is decreased. This clashes with the statistical learning point of view that states that the Bayesian posterior should provide optimal performance [Aitchison, 1975, Komaki, 1996, Zellner, 1988]. Markov Chain Monte Carlo (MCMC) methods tailored to deep learning have recently been developed and shown to have sufficient performance [Wenzel et al., 2020], implying that the cold posterior effect is not due to poor approximation methods.

Another possible cause is misspecification of the model, i.e. the assumed likelihood and prior. Due to the complexity of neural networks, it is near impossible to translate a real-world prior back into a prior over network parameters, and typically simple Gaussian priors are picked. Indeed, Fortuin et al. [2022] find in supervised learning that improving the choice of prior can reduce the cold posterior effect.

In deep reinforcement learning, misspecification of priors has been understudied. Recent methods have used Gaussian priors [Dwaracherla and Roy, 2021, Schmitt et al., 2023, Ishfaq et al., 2023, Van der Vaart et al., 2024], and the choice of prior is left as an afterthought and has not gained much attention. In our work however, we find that Gaussian priors *are* misspecified, and that improving the prior leads to more performant Bayesian deep RL algorithms.

Furthermore, another open issue is the **choice of likelihood** in RL. Supervised learning learns from ground-truth labels, often providing obvious likelihood assumptions. For example, a categorical distribution for a multi-class classification task. On the other

hand, reinforcement learning methods often learn by minimizing the difference of the current value estimate and a self-supervised (bootstrapped) value estimate of the next state, known as the temporal difference error. A likelihood on these temporal difference errors is difficult to choose correctly, as their distribution depends on the reward function, transition function and the policy itself.

Previous work in Bayesian model-free RL has typically assumed that temporal difference errors follow a normal distribution [Dearden et al., 1998, Ishfaq et al., 2023, Van der Vaart et al., 2024, Schmitt et al., 2023, Dwaracherla and Roy, 2021, Azizzadenesheli et al., 2018], likely due to ease of inference, or as a standard Bayesian extension to the squared temporal difference error that maximum likelihood approaches would use. However, as we demonstrate, this is not a realistic assumption in many benchmark tasks. In non-Bayesian RL there has been increasing interest in other losses, specifically the logistic loss [Bas-Serrano et al., 2021, Lv et al., 2024], after observing that the distribution of TD errors are closer to a logistic distribution than a normal distribution.

In this work, we establish the **existence of a cold posterior effect in Deep Q-Learning (DQN)**, and investigate potential causes. We extensively test prior and likelihood assumptions on a wide range of benchmark tasks. We find experimentally that despite widespread adoption, Gaussian priors are not a good fit in RL methods. To remedy this, we introduce Laplace priors to RL, and show that they are a better fit and provide higher performance at very low computational and implementation cost. Furthermore, we demonstrate the feasibility of meta-learning a prior on different tasks that generalizes to the test task, achieving higher performance than both Gaussian and Laplace priors. Finally, we demonstrate through statistical tests that the distribution of TD errors is neither a normal or a logistic distribution, and discuss the complications of choosing better likelihoods. Our work establishes the development of more performant priors and likelihoods as a viable future research direction to improve the performance of deep reinforcement learning algorithms.

4.2. BACKGROUND

4.2.1. REINFORCEMENT LEARNING

We consider the standard discounted infinite horizon MDP [Sutton and Barto, 2018], which is a tuple (S, A, R, T, γ) consisting of a state space S, action space A, deterministic reward function R and transition function T and discount factor $0 < \gamma < 1$.

At each time step t, the agent receives the current state $s_t \sim T(s_{t-1}, a_{t-1})$, chooses an action $a_t \sim \pi(s_t)$ from its policy π, and receives reward $r_t = R(s_t, a_t)$. The goal is to find a policy π that maximizes the expected cumulative discounted reward $\mathbb{E}[J(\pi)] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$.

Of central importance is the Q-value function $Q^\pi(s,a) = R(s,a) + \sum_{t=1}^{\infty} \gamma^t r_t$, which maps a state-action to the future expected cumulative discounted reward after executing action a in state s, and executing a given policy π afterwards.

which can also be written in a recursive relation as

$$Q^\pi(s,a) = R(s,a) + \gamma \mathbb{E}\left[Q^\pi(s',a') | s' \sim T(s,a), a' \sim \pi(s')\right] \tag{4.1}$$

4.2.2. BAYESIAN VALUE LEARNING

With a parameterized model Q_θ, Bayesian algorithms aim to infer the posterior

$$p(\theta|\mathscr{D}) = \frac{p(\mathscr{D}|\theta)\,p(\theta)}{\int p(\mathscr{D}|\theta)\,p(\theta)\,d(\theta)},$$

with $p(\mathscr{D}|\theta)$ representing the likelihood, $p(\theta)$ the prior, and $\mathscr{D}$ the observed data. The posterior, $p(\theta|\mathscr{D})$, describes the plausibility of parameter values, making it a natural way to express uncertainty.

To equip an RL agent with the ability to quantify uncertainty over values, we can construct a posterior over the parameters of a Q-function as $p(\theta|\mathscr{D}) \propto p(\mathscr{D}|\theta)\,p(\theta)$. Given that the squared error loss is proportional to the log-probability of a normal distribution, an evident choice for the likelihood in a Bayesian formulation of value-based algorithms is

$$p(\mathscr{D}|\theta) = \exp\left(-\sum_{(s,a,r,s')\in\mathscr{D}} \left[Q_\theta(s,a) - r - \gamma G(\theta,s')\right]^2\right), \tag{4.2}$$

where $G(\theta,s')$ is some estimator for the (optimal) return at s', possibly bootstrapped from our model θ. This corresponds to the assumption that temporal difference (TD) errors follow a normal distribution:

$$TD(\theta,(s,a,r,s')) \sim \mathscr{N}(0,\sigma^2). \tag{4.3}$$

Although this assumption may not be valid for every MDP, it is a convenient design decision in deep RL, and it is not surprising that various previous works have employed it [Osband et al., 2018, Schmitt et al., 2023, Dwaracherla and Roy, 2021, Azizzadenesheli et al., 2018, Ishfaq et al., 2023]. The prior is usually also chosen to be a normal distribution, which corresponds to using ℓ_2 regularisation in maximum likelihood estimation. The likelihood and prior, together with an inference method, define which posterior a Bayesian RL algorithm ends up with.

4.2.3. USING POSTERIOR DISTRIBUTIONS FOR EXPLORATION

Equipped with a posterior distribution signifying how likely a given model is, an agent can explore through a variety or techniques. Most common are optimism-based algorithms, where an attempt is made to upper bound the value for each action with some confidence level, and then explore by taking the action with the highest current upper bound. This method is widely studied in both bandit settings [Lai and Robbins, 1985] and RL [Auer et al., 2008], achieving good theoretical performance. Another approach is Thompson sampling (TS), where the posterior is sampled and the agent acts greedily with respect to the sample for one action or episode, also achieving good theoretical performance in both bandits [Agrawal and Goyal, 2012] and RL [Osband et al., 2013]. Crucially, the approximated posterior needs to be close to the true posterior for these methods to work well. For optimism, an incorrect posterior will lead to incorrect bounds. Bounds that are too loose will result in less efficiency, causing an agent to perhaps execute an action too many times before lowering the bound to a suitable level. On the other hand, bounds that are incorrect can be more catastrophic, causing an agent to never execute an action even though it might be the optimal action. Thompson sampling suffers

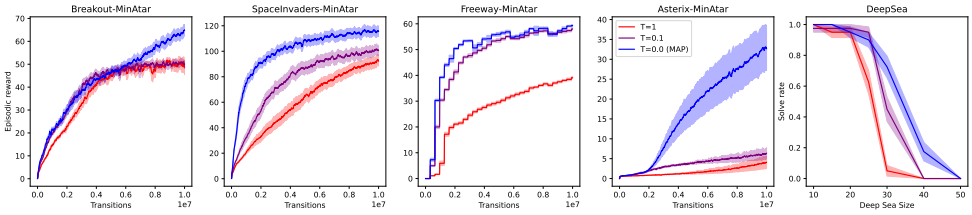

Figure 4.1: **Left four plots:** Performance of Bayesian Q-learning for three different temperatures on MinAtar. The line is the mean of 50 seeds, with shaded area displaying one standard error of the mean. There is a clear correlation between lower temperature and high performance.
Right: Solve rate of Bayesian DQN on Deep Sea at several sizes with a Gaussian prior and likelihood after 200K episodes at several temperatures. The solve rate is computed over 40 independent seeds. The shaded area displays one standard error.

from the same issues, where a posterior that does not contract fast enough leads to over-exploration, and contracting too fast leads to under-exploration. Thompson sampling with misspecified prior distributions has been studied in a bandit setting [Simchowitz et al., 2021], suffering a penalty based on the difference between the chosen prior and the true prior.

4.2.4. COLD POSTERIOR EFFECT

The cold posterior effect refers to the phenomenon where Bayesian neural networks (BNNs) sometimes achieve superior predictive performance when their posterior distributions are artificially 'cooled' by exponentiating the density by a temperature parameter $T < 1$:

$$p(\theta|\mathscr{D})^{\frac{1}{T}} \propto (p(\mathscr{D}|\theta)p(\theta))^{\frac{1}{T}} \tag{4.4}$$

As T decreases, the inverse temperature $\frac{1}{T}$ increases, effectively sharpening the posterior and therefore deliberately underestimating uncertainty. While in theory $T = 1$ is expected to be optimal [Aitchison, 1975, Komaki, 1996, Zellner, 1988], Wenzel et al. [2020] find that decreasing the temperature increases performance in deep supervised classification settings. They attribute this to misspecification of the prior distributions. On the other hand, Aitchison [2021] concludes that likelihoods are also misspecified in supervised learning, and Izmailov et al. [2021] identify data augmentation as a cause. Recently, McLatchie et al. [2024] theoretically proved under several conditions, including a large sample size, that tempering the posterior does not influence the predictive performance. Reinforcement learning with Bayesian neural networks poses an interesting point of view, since the sample size is initially small, and agents performance relies both on uncertainty quantification and predictive performance.

4.3. THE COLD POSTERIOR EFFECT IN DQN

Here, we demonstrate that Bayesian DQN methods also suffer from a cold posterior effect. In contrast to supervised learning, RL presents a unique situation, as underestimating uncertainty can lead to poor exploration during training, potentially leading to a large drop in performance. Furthermore, reinforcement learning methods are more

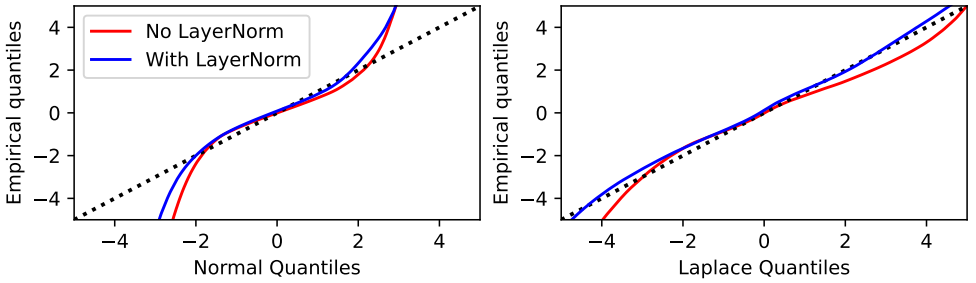

Figure 4.2: Q-Q plots with respect to a normal **(left)** or Laplace **(right)** of hidden layer weights of a Q-network after training with and without LayerNorm, aggregated over all environments. We can see that Laplace distributions are a much closer fit than normal distributions, which have flatter tails.

susceptible to misspecification of the likelihood. The Gaussian assumption on temporal difference errors is likely incorrect on many benchmarks. For example, a deterministic environment with a deterministic optimal policy will lead to deterministic temporal difference errors. Finally, while data augmentation has been used in RL [Laskin et al., 2020], it is less common and we forego any data augmentation techniques in this work.

While we should expect better performance from our Bayesian algorithm at $T = 1$, we see very clearly from our experiments that reducing the target temperature improves performance. For this experiment we ran our Bayesian Q-learning algorithm, introduced in Section 4.6.1 at $T \in \{0, 0.1, 1\}$. We tuned the hyperparameters to perform well on Breakout-Minatar at $T = 1$, and then test with the same hyperparameters for $T = 0$ and $T = 0.1$ in all MinAtar environments. The hyperparameters are optimized through Bayesian search, with details in Appendix 4.A. Figure 4.1 clearly shows that reducing the posterior temperature improves performance, even on the task for which the hyperparameters were tuned. Setting the temperature to $T = 0$ essentially reduces the sampler to maximum a posteriori (MAP) estimation, equivalent to minimizing the squared TD-loss with regularization. Furthermore, Figure 4.1 shows that even on Deep Sea [Osband et al., 2020], an exploration task, an ensemble of MAP estimates has an advantage over the posterior. In the next sections we highlight two potential problems that cause this effect: misspecified priors and misspecified likelihoods.

4.4. ARE PRIORS MISSPECIFIED?

The Bernstein von Mises' theorem states that as more data is collected, the influence of the prior will eventually fade under regularity conditions. Nonetheless, a misspecified prior leads to sub-optimal regret in bandit settings [Simchowitz et al., 2021]. While Fortuin et al. [2022] has previously concluded that priors in deep supervised learning are misspecified, this has not yet been studied in a reinforcement learning setting. We hypothesize that the typically used Gaussian priors in reinforcement learning [Dwaracherla and Roy, 2021, Schmitt et al., 2023, Van der Vaart et al., 2024] are in fact also misspecified and a likely cause for the discrepancy in performance of Bayesian DQN.

4.4.1. PRIOR MISSPECIFICATION

To test our hypothesis, we train multiple Q-learning agents and inspect the distributions of their parameters after training. Ideally, aggregating all parameters over multiple benchmark tasks would yield a distribution close to the assumed prior. Figure 4.2 shows the empirical distribution over the parameters of the second layer in a 3-layer fully connected Q-network, aggregating over 18 environments with discrete actions in the Gymnax benchmark [Lange, 2022].

We can see in the Q-Q plot that the empirical distribution over parameters is more heavy tailed than a normal distribution, signifying that a Gaussian prior might neglect certain parameter configurations that are realistic outcomes in practice. In other words, a Gaussian prior in a Bayesian DQN algorithm can actively hinder the agent from learning the true optimal values. Fortuin et al. [2022] find a similar result in classification tasks, signifying that priors for supervised learning tasks might generalize to reinforcement learning. We also plot against the quantiles of a Laplace distribution, and observe a much closer fit.

4.4.2. IMPROVING THE PRIOR

Using Layer Normalization Layer normalization [Ba et al., 2016] is a popular normalization method that normalizes the activations in a neural network, and then explicitly rescales them before applying the activation function. Recently, Gallici et al. [2024] demonstrated that layer normalization has theoretical stabilizing properties in Deep Q-learning, with visible practical advantages. Using layer normalization also provides advantages when picking a prior, by causing the outputs of a layer to be invariant under scaling of the weight matrix. As a result, the choice of prior for the weight matrix can be simplified by eliminating the need to set the scale. However, layer normalization introduces its own parameters that require a prior, and normal distributions still have an improper shape according to Figure 4.2.

Using a Laplace distribution As shown in Figure 4.2, the Laplace distribution is a much better fit to the parameter distribution that we found empirically, especially in combination with layer normalization. Therefore, simply replacing the Gaussian prior with a Laplace prior can be expected to lead to improved results. We test this hypothesis in Section 4.6.

Meta-learning a Prior While a Laplace distribution is a closer fit to the empirical distribution on the weights of the hidden layer, it is likely to be affected by the chosen activation function, position of the layer in the network, and architecture or function of the layer (e.g. convolutional, attention, layer norm). Therefore, we develop a more flexible approach for specifying priors for specific architectures. We fit a small scalar normalizing flow [Rezende and Mohamed, 2015] to the empirical distribution individually for each layer's parameters. The weights of a single layer are then assumed to be drawn i.i.d. from each layer's corresponding normalizing flow. The normalizing flow is a scalar distribution and only consists of one spline with two knots and two affine layers, causing minimal extra computational cost. We test the resulting priors in Section 4.6.

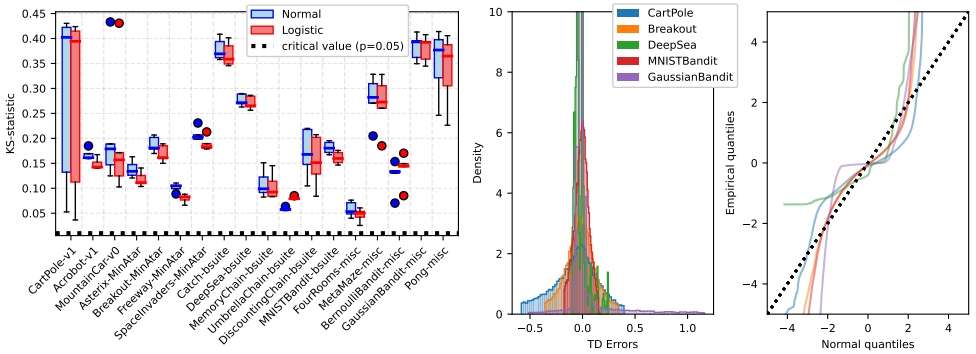

Figure 4.3: **Left:** Boxplots depicting 10 repetitions of Kolmogorov-Smirnov statistics tested against both normal and logistic distributions for TD errors of Q-learning on 19 Gymnax environments. The horizontal dashed line indicates the critical value for $p = 0.05$, which is obtained through simulation for both normal and logistic distributions. **Middle, Right**: Histograms and Q-Q plots of empirically observed temporal difference errors for Q-learning agents in 5 environments from Gymnax. The Q-Q plots are rescaled to mean 0 and standard deviation 1.

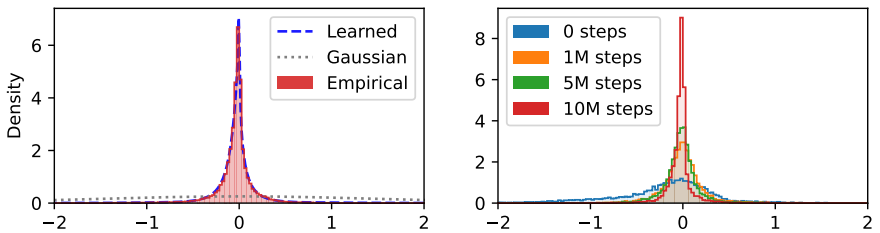

Figure 4.4: **Left:** Empirical TD errors on Breakout-MinAtar observed by a DQN agent, together with the learned likelihood model and for comparison a normal distribution with standard deviation 1.6, which is tuned for performance of our Bayesian DQN agent, but clearly does not represent the true empirical distribution well. **Right:** Empirical distributions of temporal difference errors after zero, one million, five million and ten million steps.

4.5. ARE LIKELIHOODS MISSPECIFIED?

Wrong likelihoods lead to incorrect posterior contraction and incorrect credible sets, even in the limit [Kleijn and van der Vaart, 2012]. This means that TS-style algorithms are no longer sampling actions proportional to their chance of optimality, and UCB-style algorithms are operating with incorrect bounds. Furthermore, even with infinite data, the maximum likelihood estimator under a misspecified likelihood is likely to be different from the true maximum likelihood estimator, meaning that the posterior distribution is also mispositioned.

4.5.1. GAUSSIAN LIKELIHOODS

The common choice of a Gaussian likelihood can be attributed to two reasons. First, typical DQN algorithms minimize the squared TD error, which is equivalent to maximizing

the log-density of a normal distribution:

$$\arg\min_{\theta} \sum_{(s,a,r,s')\in\mathscr{D}} \left[Q_\theta(s,a) - r - \gamma G(\theta, s')\right]^2 = \arg\max_{\theta} \sum_{\mathscr{D}} \frac{-1}{2\sigma_{TD}^2} \left[Q_\theta(s,a) - r - \gamma G(\theta, s')\right]^2 \quad (4.5)$$

$$= \arg\max_{\theta} \log p(\mathscr{D}|\theta). \quad (4.6)$$

Taking the point of view that DQN is a frequentist maximum likelihood approach that optimizes a Gaussian likelihood $p(\mathscr{D}|\theta)$, a natural practical Bayesian extension is then to pick some prior $p(\theta)$ and infer a posterior $\log p(\theta|\mathscr{D}) \propto \log p(\mathscr{D}|\theta) + \log p(\theta)$.

A more theoretical motivation for a normally distributed likelihood can be traced back to Dearden et al. [1998]. The Q-values are large sums of discounted rewards $Q^\pi = \sum_{i=1}^{\infty} \gamma^i r_i$, so by the central limit theorem [Van der Vaart, 2000] their distribution should resemble a normal distribution if the MDP is ergodic under the optimal policy. However, this argument unfortunately does not extend to our current situation, as the typical benchmarks are not ergodic and often even episodic. Furthermore, many tasks have sparse rewards where many r_i are equal to 0. Also, the r_i are not actually independent variables, since by the Markov property they are only independent when conditioning on the state s_i.

Finally, even if $Q(s,a)$ and $Q(s',a)$ for consecutive states s, s' are both normally distributed, there is no guarantee that $Q(s,a) - r - \gamma Q(s',a)$ is normally distributed because $Q(s,a)$ and $Q(s',a)$ are not independent random variables. In fact, they sum over the same future rewards r_i. Dearden et al. [1998] do make the assumption that these are independent in their Assumption 4, but also highlight that this assumption is generally false.

Recently, the logistic loss has gained some popularity for DQN [Bas-Serrano et al., 2021, Lv et al., 2024], where Lv et al. [2024] find that the logistic distribution is closer to the true error distribution than a normal distribution. In our work however, we find that neither the normal or logistic distribution is a statistically correct choice.

Empirical Validation To investigate whether our typically assumed likelihoods are valid, we train multiple Q-learning agents until convergence, and track the temporal difference errors observed at the end of training. We statistically test whether these errors come from a normal or logistic distribution using a Kolmogorov-Smirnov (KS) test [Daniel, 1990], while simultaneously estimating the parameters of the test distribution. This means that we are testing whether the TD errors come from any normal or logistic distribution. We highlight that this is a luxury that RL agents typically do not have: the likelihood scale is usually a fixed hyperparameter and has to be guessed (or tuned) correctly in advance. We refer to Appendix 4.C for details on the specific KS test that we used.

The left plot in Figure 4.3 shows that on every environment, the null hypothesis can be rejected, meaning that the TD errors follow distributions that are significantly different from both a normal and a logistic distribution. Furthermore, perhaps more troubling are the right plots in Figure 4.3, which shows that each environment in our benchmark set induces vastly different distributions for the TD errors. The Q-Q plot on the right shows that the distributions vary significantly even when correcting for the mean and

standard deviation individually per environment. This means that knowing the parameters of the likelihood ahead of time, which are usually hyper parameters of an algorithm, does not guarantee a good fit. The different shapes mean it could be very difficult to construct a single likelihood that can be expected to generalize over many tasks.

4.5.2. IMPROVING THE LIKELIHOOD

It is tempting to improve the choice of likelihood by investigating empirical TD error distributions, and pick a likelihood that is closer to the true data generating process. However, we have already seen in Figure 4.3 that the TD error distributions differ greatly per environment. Fitting one likelihood per environment is unfeasible in practice, as we are interested in the distribution of TD errors under the optimal policy. This means that fitting an empirical likelihood requires a pre-trained agent for each environment, defeating the purpose.

Nonetheless, to study whether having oracle access to this distribution would aid the agent in practice, we fit a distribution to the empirical data of each MinAtar environment, and test Bayesian DQN with the assumed likelihood. As a main potential problem, we highlight that the likelihood plays both the role of uncertainty quantification and that of a loss function. For example, Figure 4.4 shows the likelihood for Breakout-MinAtar, which has a much sharper peak. It is likely that this will cause an ill-conditioned loss landscape for gradient based optimization. Another problem when choosing a likelihood is that the distribution of actually observed temporal difference errors changes during training as shown in Figure 4.4. The wider distribution at the start can cause very large gradients under the sharply peaked likelihood in both the $T = 0$ and $T = 1$ agent. We thus hypothesize that agents with correct likelihood distributions will not necessarily do well from a performance point of view.

4.6. EMPIRICAL STUDY

We introduce our Bayesian DQN implementation, and empirically test our proposed solutions to the problems with priors and likelihoods in Bayesian DQN.

4.6.1. ALGORITHM TESTED

While several deep Bayesian Q-learning methods exist [Dwaracherla and Roy, 2021, Ishfaq et al., 2023, Van der Vaart et al., 2024, Azizzadenesheli et al., 2018, Schmitt et al., 2023], we pose that these are all special cases of the outline in Section 4.2.2. That is, they pick a return estimator $G(\theta, s')$ and a likelihood on the temporal difference error $Q_\theta(s,a) - r - \gamma G(\theta, s')$, and then use a specific inference method to approximate the posterior distribution. For example, the previously mentioned papers all take $G(\theta, s') = \max_a Q_\theta(a, s')$ and assume a normal distribution as likelihood. Schmitt et al. [2023] proposes a Laplace approximation to the posterior, whereas Dwaracherla and Roy [2021], Ishfaq et al. [2023], Van der Vaart et al. [2024], Azizzadenesheli et al. [2018] use different MCMC samplers. All these algorithms build upon the typical DQN agent.

For this work we build upon Parallel Q-learning (PQN), which is a more modern and performant baseline Q-learning algorithm. We use Watkins' Q-estimator [Watkins and Dayan, 1992], which is a small modification to PQN's Peng(λ) to accommodate for the

off-policy samples that we get from Thompson sampling. Our agent then uses

$$\log p(\theta|\mathscr{D}) \propto \sum_{(s,a,r,s')\in\mathscr{D}} \log p_{TD}(TD(\theta,s,a,r,s')|\theta) + \log p(\theta)$$

as target distribution, where p_{TD} is the likelihood of a TD error and $p(\theta)$ is the chosen prior. The likelihood is estimated from minibatches of data. We initially assume the temporal difference errors are normally distributed with standard deviation σ_{TD}, which we treat as a hyperparameter.

As inference method, we use Gradient Guided Monte Carlo (GGMC) [Garriga-Alonso and Fortuin, 2021], which is a modern MCMC sampler from a family that are known to have good performance in supervised learning [Wenzel et al., 2020]. We modify the implementation to be compatible with Optax, and translate the hyperparameters to result in equivalent learning speed of Adam. Finally, to improve the mixing of our MCMC sampler, we run an ensemble of 10 chains in parallel, making the final architecture similar to ensemble-based Q-learning methods such as BootDQN [Osband et al., 2016, 2018]. In line with prior work, we remove ϵ-greedy exploration from our agent and implement Thompson sampling by sampling one model at the start of a training batch, and acting greedily with this model for multiple steps before sampling a new one.

4.6.2. EXPERIMENTAL SETUP

Improved Priors Using the same hyperparameters for Bayesian DQN as our previous experiments, we swap out the prior with both a Laplace distribution and a Learned prior. For the Laplace distribution, we rescale the scale parameter to match the standard deviation of a normal distribution. For the learned prior we use no rescaling. The learned prior is a separate normalizing flow for each neural network layer, and is trained to fit the empirical distribution displayed in Figure 4.2 by aggregating the parameters over all environments except those of MinAtar, which we leave as testing environments similar to the typical train-test split in supervised learning.

Learned Likelihoods We fit a small normalizing flow to the temporal difference errors observed by a pre-trained PQN agent, creating a separate, environment-specific model for each MinAtar environment. These models serve as oracles that capture the empirical distribution of TD errors under a near-optimal policy. We then run our Bayesian DQN agent from scratch, but replacing the Gaussian likelihood with the density of the model of the corresponding environment.

4.6.3. NUMERICAL RESULTS

Improved Priors We can see in Figure 4.5 that improving the prior distribution can significantly improve performance of a Bayesian DQN agent. Using a Laplace prior, which is only a tiny code difference and practically no extra computational cost is already significantly better than using a normal distribution, even at the hyperparameters for which the agent with the normal prior was tuned.

Furthermore, Figure 4.5 shows that the meta-learned prior improves performance once again, almost closing the cold posterior gap in SpaceInvaders, Freeway and Asterix.

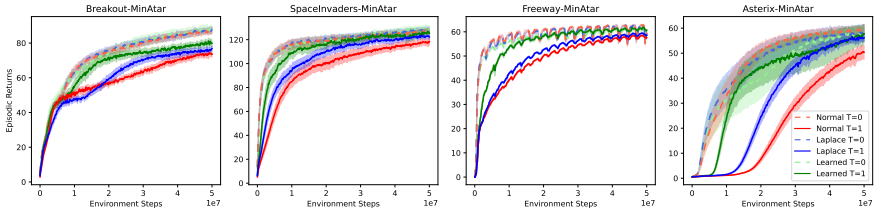

Figure 4.5: Cumulative returns of Bayesian agent with a learned prior, Laplace prior and normal prior both for $T = 1$ and $T = 0$ (MAP). Lines are the mean of 30 seeds with shaded areas denoting one standard error.

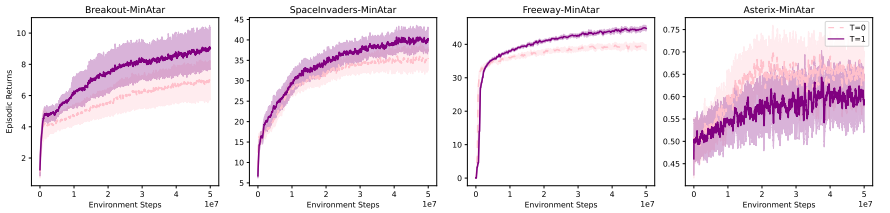

Figure 4.6: Return curves for Bayesian DQN at $T = 0$ and $T = 1$ with meta-learned prior and learned likelihoods on MinAtar. Lines are the mean of 10 independent seeds, with shaded area denoting one standard error.

The fact that this prior was fit to the neural network parameters on Gymnax environments unrelated to MinAtar highlights that it is possible to develop priors that *generalize* over environments. While this prior distribution is more involved from a programming standpoint, the computational burden is not significantly increased due to maintaining the i.i.d. assumption with neural network layers. Interestingly, improving the prior appears to have little effect on the agent with $T = 0$, indicating that the prior can aid in mitigating the cold posterior effect but does not provide better regularization in a maximum likelihood setting.

Learned Likelihoods Figure 4.6 shows the performance of Bayesian DQN with learned priors and learned likelihoods at $T = 0$ and $T = 1$. On Asterix the agent fails to learn anything, while on the other environments the agent with $T = 1$ outperforms the agent with $T = 0$. While the untempered posterior outperforms the MAP estimate in these experiments, it should be noted that all methods in this plot significantly underperform our agents where only the prior is learned. The poor results for $T = 0$ signify that the log-density of the empirical distribution leads to a poorly conditioned optimization problem, as predicted. We leave the development of likelihoods that are both realistic and easy to optimize for future research.

4.7. CONCLUSION

In this work, we have demonstrated that there exists a cold posterior effect in Deep Q-learning. We investigated possible causes by statistically testing common assumptions on the likelihoods, as well as observing empirical distribution over parameters of trained agents to evaluate choices of priors. We show empirically that making better

prior choices improves performance of Bayesian DQN agents, while keeping the performance of maximum likelihood approaches constant. Finally, we demonstrate that choosing likelihoods close to the true distribution closes the cold posterior effect. Together, our theoretical investigation and experimental results signify that there is significant room for improvement in Bayesian deep model-free RL, and more focus should be put on the likelihood and prior assumptions. We show that a principled change to a Laplace prior with a single line of code already yields significant improvements without retuning hyperparameters, and that fitting a prior to previously trained weights improves further and generalizes to new environments. A promising future research direction is to develop likelihoods that impose a smooth optimization landscape while being more realistic than the commonly assumed Gaussian.

4.A. EXPERIMENTAL DETAILS

Architecture For our Bayesian DQN agent, we used the same architecture as PQN on the Gymnax environments. We flatten the input, followed by two hidden layers of size 128 with relu activations and layernorm, and finally a linear layer with one unit for each action.

Hyperparameters We optimized the learning rate $\ell \in [5 \cdot 10^{-5}, 10^{-3}]$, standard deviation of the Gaussian prior $\sigma_p \in [10^{-2}, 10^{10}]$ and standard deviation of the Gaussian likelihood $\sigma_{TD} \in [10^{2}, 10^{-2}]$ to work well on Breakout-MinAtar at $T = 1$. The rest of the hyperparameters we keep unchanged from the default PQN hyperparameters. After 90 trials of Bayesian search we settled for the MinAtar hyperparameters displayed in Table 4.1. For Deep Sea specifically we ran an extra grid search for the likelihood standard deviation and found $\sigma_l = 0.1$ to work well environment size 20.

For our regular PQN agent experiments to empirically investigate priors and TD error distributions, we use the default hyperparameters of PQN that were tuned for Gymnax.

Priors & Likelihoods When testing Laplace priors, we did not retune any hyperparameters, but instead rescaled the scale parameter σ_p to match the standard deviation of the tuned Gaussian. For our experiments with learned priors, we simply plug in the learned prior without any rescaling. We also use unscaled learned priors for our experiments with learned likelihoods.

MCMC Sampler For the GGMC damping a and step size h parameters, we translated the default Adam [Kingma, 2014] parameter $\beta_1 = 0.9$ and our standard learning rate by $a = \exp(-(1-\beta_1))$ and $h = \sqrt{(1-\beta_1)\frac{\ell}{n_{\texttt{data}}}}$, where $n_{\texttt{data}}$ denotes the number of environment transitions the agent has observed. In contrast to Garriga-Alonso and Fortuin [2021], we fold $\sqrt{(1-\beta_1)}$ into the step size h as this more closely matched the update sizes of Adam at the same parameters. We update $n_{\texttt{data}}$ for every batch of collected trajectories, and rescale the mean likelihood of a batch by $n_{\texttt{data}}$ to reflect the full data set size.

Name	Symbol	Value
Learning rate	ℓ	10^{-3}
Prior scale	σ_p	1.679
Likelihood scale	σ_{TD}	0.56 (MinAtar), 0.1 (Deep Sea)
Ensemble size	-	10
GGMC damping	$a = \exp(-(1-\beta_1))$	$\exp(-0.1)$
GGMC step size	$h = \sqrt{(1-\beta_1)\frac{\ell}{n_{\texttt{data}}}}$	$\sqrt{\frac{10^{-4}}{n_{\texttt{data}}}}$
GGMC Preconditioner Decay	β_2	0.999

Table 4.1: Hyperparameters of our Bayesian DQN agent

4.B. NORMALIZING FLOW DETAILS

A normalizing flow is a parameterized invertable mapping $f_\psi : Z \to X$, that together with a base distribution $p_z(z)$, forms a pushforward distribution on X:

$$p_\psi(x) = p_z(f_\psi^{-1}(x))|D_{f_\psi}(f_\psi^{-1}(x))|^{-1}, \tag{4.7}$$

where $|\,.\,|$ denotes the determinant and D_{f_ψ} denotes the Jacobian of f_ψ. For more details, we refer to Rezende and Mohamed [2015]. For our work, it is most important that normalizing flows are a flexible variational inference framework, and that we can optimize the parameters ψ so that $p_\psi(x)$ matches our desired distribution. Furthermore, the invertability of f_ψ allows us to exactly evaluate the density $p_\psi(x)$ via Equation 4.7, which is crucial for our application as we want the flow to take the place of a prior or likelihood, which we need to evaluate to perform inference. Normalizing flows are typically constructed precisely so that evaluation of the log-density is cheap to compute by using operations that have simple Jacobians.

For all our normalizing flows, we used a single rational quadratic spline with two knots from the Distrax package, transforming a standard normal distribution to the target distribution. We also include affine rescaling before and after the spline. This means that each normalizing flow has only 7 (splines) + 2 (affine) + 2 (affine) = 11 parameters, making them cheap to fit and evaluate, while being much more expressive than predefined distributions.

The normalizing flow $p_\psi(x)$ is trained using Adam [Kingma, 2014] to minimize the KL-divergence between the flow and the samples

$$\arg\min_\psi KL(p_x, p_\psi) \propto \arg\min \mathbb{E}_x(-p_\psi(x)),$$

where p_x denotes the empirical distribution and x are the samples, which are neural network weights in the case of our prior experiments and TD errors in the case of our likelihood experiments.

In the experiments regarding priors, we fit an independent model to each neural network layer aggregated over all environments excluding MinAtar. Each weight in the layer shares the same normalizing flow, and weights are assumed to be drawn i.i.d. from this flow as a prior.

For the likelihood experiments, we fit the same normalizing flow architecture to the TD errors of each environment individually.

4.C. KOLMOGOROV-SMIRNOV TEST

The Kolmogorov-Smirnov (KS) test [Daniel, 1990] is a common statistical test to check whether two distributions are the same. The statistic for two cumulative density functions (cdf) F_1 and F_2, is defined as

$$D = \sup_x |F_1(x) - F_2(x)|,$$

which is the maximum deviation between the two cumulative densities. The statistic D can then be compared to a critical value D_p that depends on the significance level p to decide whether the hypothesis should be rejected, i.e. the distributions are not the same.

We apply the KS test to an empirical sample of $N = 8192$ TD errors collected by PQN *after* training for 50 million samples, and compare to both a normal distribution and a Laplace distribution. To make our tests invariant to location and scale, we normalize the TD-errors ϵ_i by

$$\hat{\epsilon}_i = \frac{\epsilon_i - \bar{\epsilon}}{\sigma_\epsilon} \tag{4.8}$$

where $\bar{\epsilon}$ and σ_ϵ are the empirical mean and standard deviation. We then construct the empirical cdf $F_{\hat{\epsilon}}$ and compute the test statistic D with respect to both the cdfs of a Gaussian and Laplace distribution with mean 0 and scale parameters 1 and $\frac{1}{\sqrt{2}}$ respectively, which corresponds to variances of 1 for both distributions

To compute the critical values for both our test statistics, we simulate D under the nulhypothesis 10000 times, each time by sampling $N = 8192$ independent samples from a Gaussian and Laplace distribution, renormalizing them equivalently to Equation 4.8, and storing the resulting KS statistics D. We then define the critical value for $p = 0.05$ as the 0.05-th percentile of our simulation results. We repeated this entire experiment 10 times for each of the 19 environments to produce the left plot in Figure 4.3.

5

CONCLUDING REMARKS

5.1. CONTRIBUTIONS

Throughout this dissertation the aim was to answer the question "Can Bayesian model-free methods be expected to outperform their non-Bayesian counterparts"?

To answer the main question, we briefly summarize the answers to the sub-questions below.

1. Can ensembles be viewed as a Bayesian approximation to the posterior? We have looked at ensembles through a Bayesian lens by employing a sequential Monte Carlo sampler as training procedure for an ensemble of Q-functions. We have experimentally seen that we outperform basic ensembles on several benchmark tasks, but perform worse than ensembles with additional tricks that promote diversity in the model on some tasks, especially those where the assumed likelihood is far from the truth. We conclude that ensembles can be viewed as Bayesian approximations when trained in a specific manner, but that this does not necessary lead to a practical advantage over other ensemble training methods.

2. Can the general structure of deep Bayesian model-free reinforcement learning methods be expected to converge?

We have proposed a unifying framework, and shown that algorithms derived from this framework approximate a general approach that converges in tabular settings. This result furthermore states that the limiting distribution only depends on the observed data, and is independent of initial starting conditions of the algorithm. Finally, we have also shown that this framework is useful for developing novel uncertainty aware algorithms by building upon PPO.

3. Are the commonly assumed likelihoods and priors realistic in typical benchmark tasks, and can they be improved?

We have Gaussian and Logistic likelihood assumptions over 18 benchmark tasks, and shown that neither likelihood is statistically valid on any of these tasks. Furthermore, we have empirically investigated the distributions over parameters of trained Q-learning agents, and show that these do not follow Gaussian distributions. We find that prior

assumptions can be improved by picking Laplace priors or meta-learning a prior distribution, but we also show that similar approaches to improve the likelihood are more complicated.

Together, the answers to our research questions highlight that Bayesian deep model-free algorithms can be similarly structured as existing non-Bayesian algorithms. Furthermore, we have shown that they are expected to converge in simplified settings, strengthening our confidence in the validity of this algorithms even in deep reinforcement learning settings. On the other hand, we have observed that typically the assumptions underlying these algorithms are incorrect, and find experimentally that it is rare for Bayesian model-free deep reinforcement learning algorithms to have an advantage over their non-Bayesian counterpart. We conclude that further research is necessary focusing on developing likelihoods and priors.

5.2. REFLECTIONS

While we have shown several appealing qualities of Bayesian model-free reinforcement learning algorithms, the question remains whether they are sensible to choose over their non-Bayesian counterparts. Our results suggest that the problem lies in the assumptions that underlie these algorithms, which are typically left as an afterthought. It appears that developing likelihoods and priors with domain knowledge for specific applications is one area where Bayesian algorithms could perform well. At the same time, our work has shown that the structure of model-free algorithms makes likelihoods difficult to develop. This suggests that Bayesian algorithms might find more success in model-based reinforcement learning, where priors and likelihoods are more easily interpretable and suffer less from misspecification issues as the MDPs data generating process itself is a valid likelihood for the collected transitions.

More generally, also outside of reinforcement learning, most recent advances in AI capabilities have stemmed in using enormous amounts of compute and data. Scaling Bayesian machine learning algorithms to this regime, perhaps by meta-learning assumptions on empirical data might also resolve the issues highlighted in this dissertation.

5.3. FUTURE WORK

This dissertation has answered several questions, but also opens up multiple interesting avenues for future research. First of all, much of this dissertation has focused on performance of the resulting algorithms, rather than pure posterior quality. Measuring the quality of the posterior is very difficult in the large parameter spaces of neural networks, but insights on the posterior quality could further explain the discrepancies between Bayesian and non-Bayesian algorithms by disentangling the problems of approximating the posterior and using the posterior.

Our second content chapter about Epistemic Bellman Operators show that model-free deep reinforcement learning algorithms can be expected to converge in tabular settings, but there is still a big gap between the tabular and deep neural network settings. While our understanding of neural networks is not yet developed enough to analyze this setting, extending Epistemic Bellman Operators to linear models and Gaussian process

models could increase the impact of the theory, and perhaps eventually be extended to a simplified neural network setting through Neural Tangent Kernel theory.

Furthermore, our third content chapter shows that likelihood and prior assumptions are violated on benchmark tasks in practice. While we already offer a few practical solutions, further research into priors and likelihoods is warranted. On the priors side, future work should attempt to meta-learn priors over several network architectures to generalize to both new architectures and new tasks. Furthermore, an interesting experiment that exploits the amount of compute available today is to meta-learn such priors on pre-trained network weight from previously published work with open source models. On the side of likelihoods it would be interesting to try to learn the likelihood on-line through density estimation methods such as normalizing flows.

Once misspecification issues are sufficiently resolved, it makes sense to go back to developing cheaper and simpler Bayesian approximation methods that still remain accurate. The holy grail in this direction would be a Bayesian sampler that can be dropped in place of popular optimizers such as Adam and that performs well.

5.4. FINAL REMARKS

In this dissertation we worked towards understanding Bayesian model-free deep reinforcement learning algorithms. We have found several desirable properties such as their convergence in simplified settings, but also some main issues in their application in practice. While there is not yet clear evidence that Bayesian model-free reinforcement learning algorithms should be expected to outperform tradition deep reinforcement learning algorithms, they remain an interesting problem to study and there is plenty of future work to possibly improve their effectiveness.

BIBLIOGRAPHY

S. Agrawal and N. Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on learning theory*, pages 39–1. JMLR Workshop and Conference Proceedings, 2012.

S. Agrawal and N. Goyal. Near-optimal regret bounds for thompson sampling. *Journal of the ACM (JACM)*, 64(5):1–24, 2017.

J. Aitchison. Goodness of prediction fit. *Biometrika*, 62(3):547–554, 1975.

L. Aitchison. A statistical theory of cold posteriors in deep neural networks. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=Rd138pWXMvG`.

P. Auer, T. Jaksch, and R. Ortner. Near-optimal regret bounds for reinforcement learning. *Advances in neural information processing systems*, 21, 2008.

K. Azizzadenesheli, E. Brunskill, and A. Anandkumar. Efficient exploration through bayesian deep q-networks. In *2018 Information Theory and Applications Workshop (ITA)*. IEEE, 2018.

L. J. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016. URL `http://arxiv.org/abs/1607.06450`.

C. Bai, L. Wang, L. Han, J. Hao, A. Garg, P. Liu, and Z. Wang. Principled exploration via optimistic bootstrapping and backward induction. In *International Conference on Machine Learning*, 2021.

J. Bas-Serrano, S. Curi, A. Krause, and G. Neu. Logistic q-learning. In *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 3610–3618. PMLR, 2021. URL `http://proceedings.mlr.press/v130/bas-serrano21a.html`.

M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

M. G. Bellemare, W. Dabney, and R. Munos. A distributional perspective on reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2017.

G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI gym, 2016.

T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. In *International Conference on Learning Representations*, 2018.

A. Cabezas, A. Corenflos, J. Lao, and R. Louf. Blackjax: Composable Bayesian inference in JAX, 2024.

M. Cai, M. Del Negro, E. Herbst, E. Matlin, R. Sarfati, and F. Schorfheide. Online estimation of dsge models. *The Econometrics Journal*, 24(1):C33–C58, 2021.

T. Chen, E. Fox, and C. Guestrin. Stochastic gradient hamiltonian monte carlo. In *International Conference on Machine Learning*, 2014.

K. Ciosek, Q. Vuong, R. Loftin, and K. Hofmann. Better exploration with optimistic actor critic. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

F. D'Angelo and V. Fortuin. Repulsive deep ensembles are bayesian. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

W. Daniel. *Applied Nonparametric Statistics*. Duxbury advanced series in statistics and decision sciences. PWS-KENT Pub., 1990.

H.-D. Dau and N. Chopin. Waste-free sequential monte carlo. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 84(1):114–148, 2022.

E. Daxberger, A. Kristiadi, A. Immer, R. Eschenhagen, M. Bauer, and P. Hennig. Laplace redux-effortless bayesian deep learning. In *Advances in Neural Information Processing Systems*, 2021.

R. Dearden, N. Friedman, and S. Russell. Bayesian Q-learning. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference*. AAAI Press / The MIT Press, 1998.

J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de Las Casas, et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897), 2022.

P. Del Moral, A. Doucet, and A. Jasra. Sequential monte carlo samplers. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(3):411–436, 2006.

T. G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15. Springer Berlin Heidelberg, 2000.

A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

V. Dwaracherla and B. V. Roy. Langevin dqn, 2021.

A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. J. R. Ruiz, J. Schrittwieser, G. Swirszcz, D. Silver, D. Hassabis, and P. Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610 (7930), 2022. doi: 10.1038/s41586-022-05172-4.

M. Fellows, K. Hartikainen, and S. Whiteson. Bayesian bellman operators. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

V. Fortuin, A. Garriga-Alonso, S. W. Ober, F. Wenzel, G. Ratsch, R. E. Turner, M. van der Wilk, and L. Aitchison. Bayesian neural network priors revisited. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=xkjqJYqRJy`.

M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg. Noisy networks for exploration, 2019.

Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, 2016.

M. Gallici, M. Fellows, B. Ellis, B. Pou, I. Masmitja, J. N. Foerster, and M. Martin. Simplifying deep temporal difference learning. *CoRR, abs/2407.04811*, 2024.

A. Garriga-Alonso and V. Fortuin. Exact langevin dynamics with stochastic gradients. *arXiv:2102.01691*, 2021.

M. Geiger, A. Jacot, S. Spigler, F. Gabriel, L. Sagun, S. d'Ascoli, G. Biroli, C. Hongler, and M. Wyart. Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(2), 2020.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*. PMLR, 2018.

A. Hannun, C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

E. Hüllermeier and W. Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

H. Ishfaq, Q. Lan, P. Xu, A. R. Mahmood, D. Precup, A. Anandkumar, and K. Azizzadenesheli. Provable and practical: Efficient exploration in reinforcement learning via langevin monte carlo, 2023.

P. Izmailov, S. Vikram, M. D. Hoffman, and A. G. G. Wilson. What are bayesian neural network posteriors really like? In *International conference on machine learning*, pages 4629–4640. PMLR, 2021.

D. P. Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

B. Kleijn and A. van der Vaart. The Bernstein-Von-Mises theorem under misspecification. *Electronic Journal of Statistics*, 6:354–381, 2012. doi: 10.1214/12-EJS675. URL https://doi.org/10.1214/12-EJS675.

F. Komaki. On asymptotic properties of predictive distributions. *Biometrika*, 83(2):299–313, 1996. URL http://www.jstor.org/stable/2337602.

T. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in Applied Mathematics*, 6(1):4–22, 1985. ISSN 0196-8858. doi: https://doi.org/10.1016/0196-8858(85)90002-8. URL https://www.sciencedirect.com/science/article/pii/0196885885900028.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

R. T. Lange. gymnax: A JAX-based reinforcement learning environment library. http://github.com/RobertTLange/gymnax, 2022. Accessed: 2024-08-20.

R. T. Lange and H. Sprekeler. Learning not to learn: Nature versus nurture in silico. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence (AAAI-22)*, volume 36, 2022.

M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895, 2020.

K. Lee, M. Laskin, A. Srinivas, and P. Abbeel. Sunrise: A simple unified framework for ensemble learning in deep reinforcement learning. In *International Conference on Machine Learning*, 2021.

Q. Liu and D. Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

F. Llorente, L. Martino, J. Read, and D. Delgado-Gómez. Optimality in noisy importance sampling. *Signal Processing*, 194:108455, 2022.

O. Lockwood and M. Si. A review of uncertainty for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 2022.

C. Lu, J. Kuba, A. Letcher, L. Metz, C. Schroeder de Witt, and J. Foerster. Discovered policy optimisation. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

O. Lv, B. Zhou, and L. F. Yang. Modeling bellman-error with logistic distribution with applications in reinforcement learning. *Neural Networks*, 177:106387, 2024.

A. Mandhane, A. Zhernov, M. Rauh, C. Gu, M. Wang, F. Xue, W. Shang, D. Pang, R. Claus, C.-H. Chiang, C. Chen, J. Han, A. Chen, D. J. Mankowitz, J. Broshear, J. Schrittwieser, T. Hubert, O. Vinyals, and T. A. Mann. MuZero with self-competition for rate control in VP9 video compression, 2022. URL `https://api.semanticscholar.org/CorpusID:246823228`.

D. J. Mankowitz, A. Michi, A. Zhernov, M. Gelmi, M. Selvi, C. Paduraru, E. Leurent, S. Iqbal, J.-B. Lespiau, A. Ahern, T. Koppe, K. Millikin, S. Gaffney, S. Elster, J. Broshear, C. Gamble, K. Milan, R. Tung, M. Hwang, T. Cemgil, M. Barekatain, Y. Li, A. Mandhane, T. Hubert, J. Schrittwieser, D. Hassabis, P. Kohli, M. Riedmiller, O. Vinyals, and D. Silver. Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, 618 (7964), 2023. doi: 10.1038/s41586-023-06004-9.

Y. McLatchie, E. Fong, D. T. Frazier, and J. Knoblauch. Predictive performance of power posteriors. *arXiv preprint arXiv:2408.08806*, 2024.

T. Miconi, A. Rawal, J. Clune, and K. O. Stanley. Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. In *International Conference on Learning Representations*, 2018.

V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015. ISSN 00280836.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*. PMLR, 2016.

R. M. Neal. *Bayesian Learning for Neural Networks*. Lecture Notes in Statistics. Springer, New York, NY, 1996.

R. M. Neal et al. Mcmc using hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*. Chapman and Hall/CRC, 2011.

I. Osband, D. Russo, and B. Van Roy. (more) efficient reinforcement learning via posterior sampling. *Advances in Neural Information Processing Systems*, 26, 2013.

I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

I. Osband, J. Aslanides, and A. Cassirer. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

I. Osband, Y. Doron, M. Hessel, J. Aslanides, E. Sezener, A. Saraiva, K. McKinney, T. Lattimore, C. Szepesvari, S. Singh, B. V. Roy, R. Sutton, D. Silver, and H. V. Hasselt. Behaviour suite for reinforcement learning. In *International Conference on Learning Representations*, 2020.

G. Ostrovski, M. G. Bellemare, A. Oord, and R. Munos. Count-based exploration with neural density models. In *International Conference on Machine Learning*, 2017.

L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. In *Advances in neural information processing systems*, volume 35, 2022.

B. O'Donoghue, I. Osband, R. Munos, and V. Mnih. The uncertainty bellman equation and exploration. In *International Conference on Machine Learning*, 2018.

T. Papamarkou, M. Skoularidou, K. Palla, L. Aitchison, J. Arbel, D. Dunson, M. Filippone, V. Fortuin, P. Hennig, J. M. Hernández-Lobato, et al. Position: Bayesian deep learning is needed in the age of large-scale ai. In *International Conference on Machine Learning*, pages 39556–39586. PMLR, 2024.

D. Rezende and S. Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, 2015.

D. Russo and B. Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

D. Russo and B. Van Roy. An information-theoretic analysis of thompson sampling. *Journal of Machine Learning Research*, 17(68):1–30, 2016.

S. Schmitt, J. Shawe-Taylor, and H. van Hasselt. Exploration via epistemic value estimation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, 2023.

J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839), 2020. ISSN 1476-4687. doi: 10.1038/s41586-020-03051-4.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017.

M. Simchowitz, C. Tosh, A. Krishnamurthy, D. J. Hsu, T. Lykouris, M. Dudik, and R. E. Schapire. Bayesian decision-making under misspecified priors with applications to meta-learning. *Advances in Neural Information Processing Systems*, 34:26382–26394, 2021.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2), 1999.

W. R. Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

A. Van der Vaart. *Asymptotic Statistics*, volume 3. Cambridge University Press, 2000.

P. R. Van der Vaart, N. Yorke-Smith, and M. T. J. Spaan. Bayesian ensembles for exploration in deep reinforcement learning. In *Proceedings of the 2024 International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '24, Richland, SC, 2024. International Foundation for Autonomous Agents and Multiagent Systems.

H. Van Hasselt. Double q-learning. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL `https://proceedings.neurips.cc/paper_files/paper/2010/file/091d584fced301b442654dd8c23b3fc9-Paper.pdf`.

H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn, 2016.

C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

F. Wenzel, K. Roth, B. Veeling, J. Swiatkowski, L. Tran, S. Mandt, J. Snoek, T. Salimans, R. Jenatton, and S. Nowozin. How good is the bayes posterior in deep neural networks really? In *International Conference on Machine Learning*, 2020.

T. Yang, H. Tang, C. Bai, J. Liu, J. Hao, Z. Meng, P. Liu, and Z. Wang. Exploration in deep reinforcement learning: From single-agent to multiagent domain. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

K. Young and T. Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments, 2019.

A. Zellner. Optimal information processing and bayes's theorem. *The American Statistician*, 42(4):278–280, 1988.

ACKNOWLEDGEMENTS

This dissertation would not have been possible without the support and guidance of many people in both my professional and personal lives, which have happily grown closer together over these past years. During my years in Delft I have become close friends with colleagues and while I can not thank everyone in this acknowledgement, I am grateful for all the interesting conversations we have had and will forever cherish these past years.

First and foremost I would like to thank my supervisors Matthijs and Neil, for being very dependable supervisors with countless invaluable insights for both career and life. I will forever remember the words "A PhD is a marathon, not a sprint", which were spoken after I announced I made zero progress in the preceding week. In all seriousness, your guidance has had a huge positive impact on my academic career, and you were excellent at providing the support I needed during this PhD. This includes a few encouragements to make an earlier deadline, helping me with writing, or discussing my ideas during our meetings. Thank you for your support and trust, and letting me work on whatever I deemed interesting and useful. This dissertation and its contents would not have been possible without you.

Secondly, I thank the promotion committee, Frans, Joris, Nils and Vincent, for their time and careful reading of my work. I appreciate their feedback, which has improved the final dissertation.

Frans, the Bayesian RL lecture we prepared together was a great learning experience and I appreciate the teaching opportunity. It is unfortunate that the Postdoc position fell through as I was excited to work together on control as inference, but everything worked out fine in the end. Thank you for your interest in my work and encouraging words, which have really increased my academic confidence.

Wendelin, thank you for the insightful discussions at both the weekly meetings and the reading group, and of course for the infinite stream of suggested reading.

Thank you Yaniv, for making the lab a more collaborative and social place. Thank you for the effort you put into getting our group of friends together, as you still do to this day. Having close friends in our office has had a truly positive impact on both my enjoyment and academic output. Thank you also for the endless discussions about endless topics, which I hope we will continue to have far into the future.

Moritz, thank you for your the interesting academic discussions and collaborations we have had, but foremost I should thank you for being a great friend and inviting us so often to your home. I hope we will continue this friendship in the future. Also many thanks for often taking my side in discussions with Yaniv when he is obviously wrong, but especially when he might actually be right.

Thank you Joery, for our chats about probabilistic machine learning and variational inference, but of course also for our JAX debugging discussions. Together we have truly

overcome some deeply cryptic errors and I appreciate that I could ask you whether it was just me.

Junhan and Caroline, thank you for greatly contributing the social fabric of office E4.260. I hope we will continue to have barbecue, hotpot or pizza dinners with the office.

Thank you Noah, Kim, Moritz, Joery, and Max, for those initial months in office W2.540. Being lost in a new PhD at the same time was a very unique experience that I am happy I could share with you.

Thank you Mert, Thiago and Jinke for our chats and your career advice.

Oussama and Davide, thanks for a wonderful time at UAI and our amicable chats at the office.

Thank you Laurens, for our chats at the office but of course also your support in the top lane.

Thank you Tristan, for spontaneously letting me tag along on your campervan adventure through New Zealand after meeting me for the first time at AAMAS'24. The entire trip was truly a life changing experience for me.

Andreea, thank you for our chats at the office and taking over the organization of the weekly meetings.

Thank you Daan, David, IJsbrand, Koen, Jeroen and Marten for becoming my closest group of friends over the duration of my PhD. I truly appreciate that you were there for me in both good and tough times.

Thanks Lucas and Quinten for those years at the Bloemstraat and our continued friendship.

Thank you Aad, Maryse, Marianne and Jacob for your unconditional support. You have all inspired me to go on this path and I am very grateful for that.

Merel, thank you for your endless positivity and enthousiasm when it comes to my ability and work. The past (almost) two years you have had an immeasurable impact on this dissertation and my life. You inspire me.