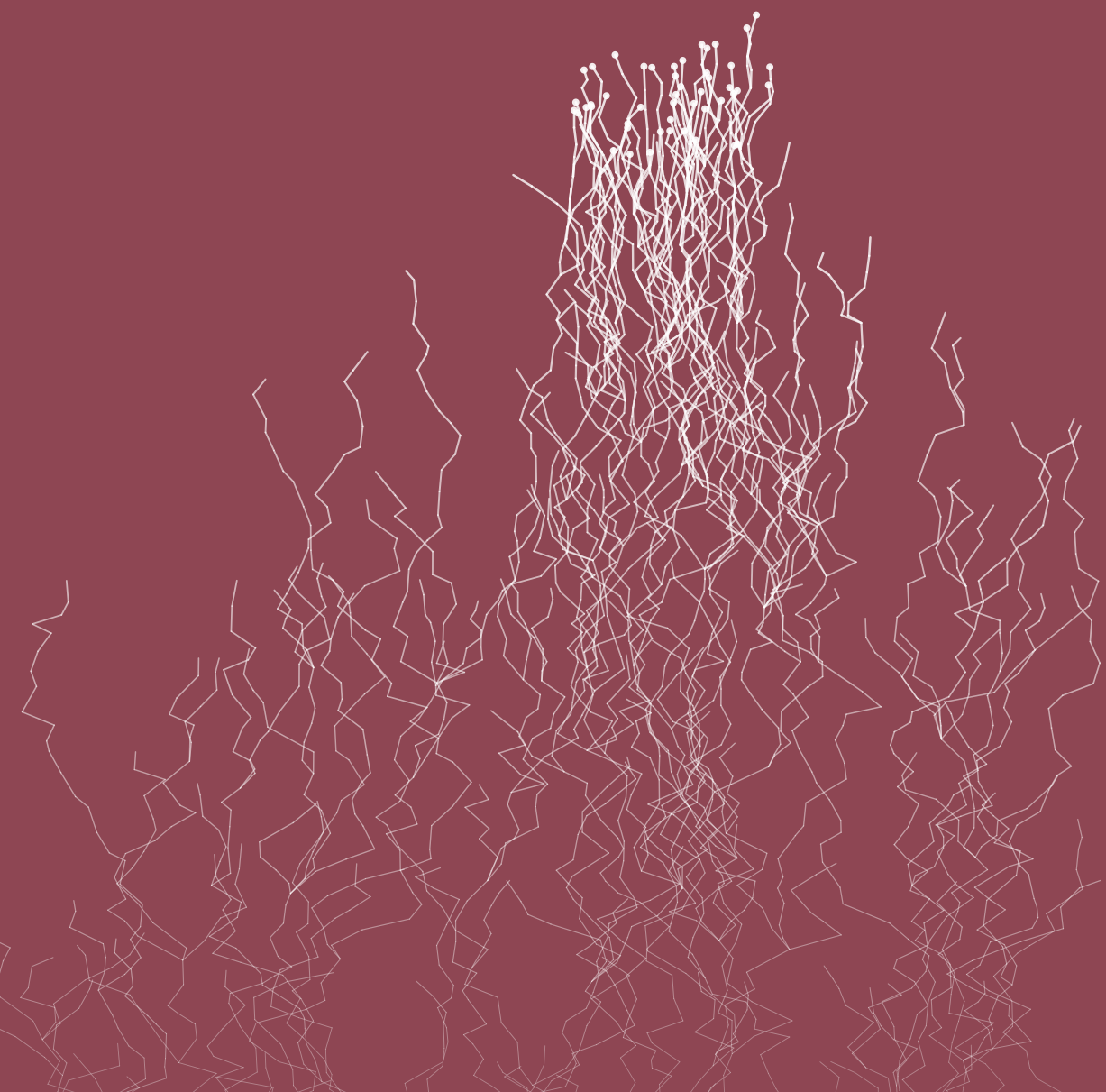
Approximate Bayes-Optimal Reinforcement Learning and Planning

Joery A. de Vries



Propositions

accompanying the dissertation

Approximate Bayes-Optimal Reinforcement Learning and Planning

by

Joery A. DE VRIES

1. Uncertainty estimates cannot inform decision making when they are miscalibrated (Chapter 2).
2. As research progresses, algorithmic methods coalesce to the point of indistinguishability (Chapter 3).
3. Inference within a well specified model outperforms any heuristics (Chapter 4).
4. Despite the current information age, scientists still live on information islands.
5. Computation for planning induces 'inaction' that exhausts time and resources, creating an economic trade-off against random guessing, which incurs costs solely within the environment.
6. AI scientists publish fast but think slow.
7. AI scientists work towards their own replacement.
8. Experimental design in deep learning seldom leads to insightful conclusions.
9. Advances in machine learning are driven more by infrastructure than algorithm-engineering.
10. In science, individual contributions are noisy and incomplete yet collectively there is progress.

These propositions are regarded as opposable and defendable, and have been approved as such by the promotors prof. dr. M. T. J. Spaan and prof. dr. M. M. de Weerdt.

Approximate Bayes-Optimal Reinforcement Learning and Planning

Approximate Bayes-Optimal Reinforcement Learning and Planning

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
Monday, 5 October 2026 at 15:00

by

Joery Ariën de Vries

This dissertation has been approved by the promoters.

Composition of the doctoral committee:

Rector Magnificus,	chairperson
Prof.dr. M.T.J. Spaan,	Delft University of Technology, promotor
Prof.dr. M.M. de Weerdt,	Delft University of Technology, promotor
Independent members:	
Prof.dr.ir. M.J.T. Reinders,	Delft University of Technology
Prof.dr. A. Plaat,	Leiden University
Prof.dr. M. Toussaint,	Technische Universität Berlin
Prof.dr. S. Trimpe,	RWTH Aachen University
Prof.dr. J. Alonso-Mora,	Delft University of Technology, *reserve member*
Non-independent members:	
Dr. S. Bierman,	DSM-Firmenich

This dissertation is supported by the AI4b.io program, a collaboration between TU Delft and dsm-firmenich, which is fully funded by dsm-firmenich and the RVO (Rijksdienst voor Ondernemend Nederland).



Keywords: Deep Reinforcement Learning, Bayesian Optimization, Uncertainty

Cover by: Joery A. de Vries

Copyright: 2026 by J.A. de Vries

An electronic version of this dissertation is available at: http://repository.tudelft.nl/.

Preface

This dissertation explores and develops algorithms for approximate optimal exploration in sequential decision making tasks, with the overencompassing ambition to develop stronger algorithms for optimal experiment design in autonomous laboratories. Fortunately, a lot of theoretical groundwork already exists for this problem, this gives us a known, albeit infeasible, upper-bound on the performance of our algorithms. Thus, a main theme throughout the dissertation is to improve *scaling* of approximate algorithms that directly target this theoretical ideal. That is, 1) are our approximations consistent with our desired target? And 2) how do we make better use of our available computation resources to get an approximate solution faster, or at reduced costs?

Because this dissertation was developed in an interdisciplinary research lab, it targets two main audiences. The technical chapters are primarily written for machine learning and reinforcement learning researchers, while the introduction, discussion, and summary sections are written for a broader audience.

Contents

Preface i

Summary vii

Samenvatting ix

Acronyms xi

1 Introduction 1

1.1 Abstracting Sequential Experiment Design . . . 3

1.1.1 Optimal Experiment Design . . . 3

1.1.2 Sequential Design as Numerical Optimization . . . 4

1.1.3 An Agent-Based Abstraction for Dynamic Control . . . 6

*Summary . . . 8

1.2 Scope: Challenges in Scaling Bayesian Search . . . 8

1.2.1 Semi-Amortization for Generalization . . . 10

1.2.2 Scalable (Approximate) Bayesian Planning . . . 11

1.3 Aim and Research Goals . . . 13

1.4 Outline . . . 15

2 Posterior Uncertainty in Meta-Trained Agents 17

2.1 Introduction . . . 18

2.2 Related Work . . . 19

2.3 Preliminaries . . . 20

2.3.1 Inference in Meta-RL . . . 20

2.4 Laplace Variational Recurrent Networks . . . 23

2.5 Experimental Validation . . . 25

2.5.1 Supervised Learning . . . 26

2.5.2 Reinforcement Learning . . . 27

2.6 Conclusions . . . 29

2.A Derivations . . . 30

2.A.1 Lower Bounds . . . 30

2.A.2 Laplace Variational Recurrent Model . . . 33

2.B Implementation Details . . . 37
2.B.1 Model Architecture and Optimization . . . 37
2.B.2 Variational Posterior Baseline . . . 38
2.B.3 Predictive Ensemble Averaging . . . 38
2.B.4 Environment Design . . . 39
2.B.5 Experimental Design and Hyperparameters . . . 40
2.C Supplementary Results . . . 42

3 Efficient Semi-Amortized Policy Inference 47
3.1 Introduction . . . 48
3.2 Background . . . 49
3.2.1 Control as Inference . . . 49
3.2.2 Particle Filter Planning for RL . . . 50
3.3 Tailoring Particle Filter Planning to RL . . . 52
3.3.1 Adapting the Proposals for Sample-Efficiency . . . 53
3.3.2 Handling Terminal States with Revived Resampling . . . 54
3.3.3 Mitigating Path Degeneracy for Policy Inference . . . 55
3.3.4 Search-Based Values for Learning Targets . . . 56
3.4 Experiments . . . 57
3.4.1 Main Results . . . 58
3.4.2 Ablations . . . 58
3.5 Related Work . . . 60
3.6 Conclusion . . . 61
3.A Derivations . . . 62
3.A.1 Lower-bound . . . 62
3.A.2 Regularized Policy Improvement . . . 63
3.A.3 Proof of Theorem 3.1 . . . 63
3.A.4 Importance sampling weights . . . 63
3.B Experiment Details . . . 64
3.B.1 Environments . . . 64
3.B.2 Hardware Requirements . . . 65
3.B.3 Hyperparameters, Model Training, and Software Versioning . . . 66
3.B.4 Neural Network Architectures . . . 67
3.B.5 Details on Constrained Proposals . . . 70
3.B.6 Details on Figure 3.1 . . . 70
3.C Pseudocode . . . 70
3.D Supplementary Results . . . 72

4 Scalable Approximate Bayes-Adaptive Monte-Carlo Planning **75**
4.1 Introduction . . . 76
4.2 Background . . . 78
4.3 Bayes-Adaptive Particle-Filter Planning . . . 82
4.3.1 Improved E-step with Belief-SMC planning . . . 82
4.3.2 Gradient Based (EM)-step for Amortized Variational Bayes . . . 84
4.4 Experiments . . . 86
4.5 Related Work . . . 88
4.6 Conclusion . . . 89
4.A Derivations & Proofs . . . 89
4.B Control as Inference Details . . . 91
4.B.1 On Determinstic Rewards in Control-as-Inference . . . 91
4.C Experiment Details . . . 92
4.C.1 Environments . . . 92
4.C.2 Hardware Requirements . . . 93
4.C.3 Hyperparameters, Model Training, and Software Versioning . . . 93
4.C.4 Neural Network Architectures . . . 94

5 Discussion **97**
5.1 Research Findings . . . 97
5.2 Outlook . . . 100
5.2.1 Limitations and Future Work . . . 101
5.2.2 Moving to Automated Designs in Real Laboratories . . . 102
5.3 Conclusion . . . 104

Bibliography **107**

Dankwoord **127**

Curriculum Vitæ **129**

List of Publications **131**
List of Repositories . . . 131

Summary

This dissertation studies the optimal decision making under uncertainty problem from the context of sequential experiment planning in autonomous laboratories. Our focus is on deep reinforcement learning methods, which we first motivate by building up from the challenges involved in conventional optimal experiment design methods and black-box numerical optimization. Secondly, we frame deep reinforcement learning in a unifying view of commonplace Bayesian optimization methods to conceptually streamline extensions or adoption of reinforcement learning methods in existing infrastructure. Each chapter investigates challenges and questions that emerge from our framing, which more specifically relate to uncertainty quantification and proper algorithm-hardware tailoring for improving performance scaling.

Bayesian inference of large complex models is typically computationally infeasible unless stringent, but possibly unrealistic, assumptions are placed on the model of the physical system that we're trying to capture. Deep learning has since emerged as a powerful way to approximate this computationally infeasible routine through pattern recognition based on an abundance of data. This principle underpins most of the success stories of recent artificial intelligence breakthroughs, like agentic chat-bots or game-playing agents. This dissertation investigates this principle from a more fundamental basis to improve our algorithmic tools for the overarching experiment-design use-case.

The field of meta-reinforcement learning, similarly, has often used deep learning to capture various sources of uncertainty for decision-making agents, such as unknown rewards or world dynamics. This approximates what is known as Bayes-optimal control, an agent that optimally trades-off between information seeking actions, and actions that maximize a (subjective) value. In Chapter 2 we show how common deep learning approaches, like recurrent neural networks, for meta-reinforcement learning do not appropriately capture true uncertainty from the Bayesian viewpoint. Rather, we found that uncertainty is obfuscated in the activations of the neural networks. We present the Laplace Variational Recurrent Neural Network to extract this implicit uncertainty as a means to enable practitioners to deal with uncertainty, explicitly.

From Chapter 2 we built the insight that one of the roles that deep learning plays in deep reinforcement learning is to amortize an exponentially costly policy optimization problem. A well-known consequence of amortization is that we always have to deal with an amortization gap (approximation gap) between the true values and the predicted values. Improving the performance of an algorithm, thus, requires a choice of tools that minimize the runtime and data needed to reduce this gap. Among the most successful approaches in this space are those that use adaptive computation at inference time to reduce the amortization gap and complement the neural network. Monte-Carlo tree search, for instance, achieved strong data-efficiency and enabled breakthroughs like defeating the human world champion in Go. However, Monte-Carlo tree search based methods do not

favorably scale in runtime. Thus, in Chapter 3 we improve the state-of-the-art of sequential Monte-Carlo planners, which permit much more efficient parallelization on modern GPU hardware. Our newly proposed algorithm demonstrated improved scaling in performance given additional planning budget, not only in data-efficiency but also in runtime-efficiency compared to prior approaches.

The final Chapter 4 exploits our finding that sequential Monte-Carlo based planning can improve runtime complexity by shifting computational budget from neural network learning, to additional planning budget to get higher quality data. Since in meta-learning settings we typically deal with learning from trajectory-based data, we often find that when trajectories increase in length, any updates to a neural network architecture will become more expensive. This is because the computational cost of gradient updates grows with trajectory length, which causes bottlenecks in scaling. For instance, updates can start to exceed hardware memory limits, forcing sequential processing of smaller batches and reducing throughput. Thus, we again find that the performance scaling to additional planning-compute in deep meta-reinforcement learning can be much improved when equipped with our designed sequential Monte-Carlo planner. In contrast to prior approaches, our planner explicitly accounts for how uncertainty evolves in light of new data. As a result, our method directly targets an optimal policy that trades-off exploration with exploitation, without *ad-hoc* exploration heuristics.

In summary, this dissertation advances the state of probabilistic reinforcement learning for uncertainty-based decision-making. We show that Bayesian optimization and deep reinforcement learning can be formulated very similarly from a probabilistic viewpoint, and that only (relatively-) minor assumptions and choice of approximation tools shift us from one setting to the other. This is a particularly useful insight for our use-case of self-driving laboratories, since at the time of writing, Bayesian optimization is a more established approach in practice. Our findings establish a stronger common ground between the seemingly separate Bayesian optimization and reinforcement learning fields, and provide a principled basis for transferring approximation tools between these communities.

Samenvatting

Dit proefschrift bestudeert het probleem van optimale besluitvorming onder onzekerheid vanuit de context van sequentiële experimentplanning in autonome laboratoria. Onze focus ligt op deep reinforcement learning methoden, die we eerst motiveren vanuit de uitdagingen van conventionele optimale experimentontwerp-methoden en black-box numerieke optimalisatie. Vervolgens kaderen we deep reinforcement learning in een gecombineerd perspectief met haalbare Bayesiaanse optimalisatiemethoden, om conceptueel de uitbreiding of adoptie van reinforcement learning methoden in bestaande infrastructuur te stroomlijnen. Elk hoofdstuk onderzoekt uitdagingen en vragen die voortkomen uit ons raamwerk, deze hebben specifiek betrekking op onzekerheidsquantificatie en een juiste afstemming tussen algoritme-en-hardware ter verbetering van prestatieschaling.

Bayesiaanse herleiding van grote complexe modellen is doorgaans computationeel onhaalbaar, tenzij strenge, maar mogelijk onrealistische, aannames worden gemaakt over het model van het fysische systeem dat we proberen te beschrijven. Deep learning is sindsdien ontwikkeld tot een krachtig gereedschap om deze computationeel onhaalbare routine te benaderen door middel van patroonherkenning op basis van een overvloed aan data. Dit principe heeft zich vertaalt in de meeste recente successen en doorbraken in kunstmatige intelligentie, zoals agentische chatbots of game-playing agents. Dit proefschrift onderzoekt dit principe op een meer fundamentele basis om onze algoritmische gereedschappen te verbeteren voor de overkoepelende experimentontwerp-toepassing.

Het veld van meta-reinforcement learning heeft op vergelijkbare wijze vaak deep learning ingezet om verschillende bronnen van onzekerheid voor besluitvormende agenten te vangen, zoals onbekende beloningen of werelddynamica. Dit benadert wat bekend staat als Bayes-optimale controle, een agent die optimaal afweegt tussen informatiezoekende acties en acties die een (subjectieve) waarde maximaliseren. In Hoofdstuk 2 laten we zien hoe benaderingen van deep learning op recurrente neurale netwerken, voor meta-reinforcement learning, de werkelijke onzekerheid vanuit een Bayesiaans oogpunt niet adequaat opvangen. We constateerden juist dat onzekerheid verborgen zit in de activaties van de neurale netwerken. We presenteren het Laplace Variational Recurrent Neural Network om deze impliciete onzekerheid te extraheren, als middel om gebruikers in staat te stellen expliciet met onzekerheid om te gaan.

Uit Hoofdstuk 2 bouwen we het inzicht op dat een van de rollen die deep learning speelt in deep reinforcement learning het amortiseren van een exponentieel kostbaar beleidsoptimalisatieprobleem omvat. Een bekend gevolg van amortisatie is dat we altijd te maken hebben met een amortisatiekloof (benaderingskloof) tussen de werkelijke waarden en de voorspelde waarden. Het verbeteren van de prestaties van een algoritme vereist dus een keuze van gereedschappen die de rekentijd en data minimaliseren die nodig zijn om deze kloof te verkleinen. Tot de meest succesvolle benaderingen in dit domein behoren methoden die adaptieve berekening tijdens inferentie inzetten om de amortisatiekloof

te verkleinen en het neurale netwerk aan te vullen. Monte-Carlo tree search bereikte bijvoorbeeld sterke data-efficiëntie en maakte doorbraken mogelijk zoals het verslaan van de menselijke wereldkampioen in Go. Echter, op Monte-Carlo tree search gebaseerde methoden schalen niet gunstig in rekentijd. Daarom verbeteren we in Hoofdstuk 3 de state-of-the-art van sequentiële Monte-Carlo planners, die veel efficiëntere parallellisatie op moderne GPU-hardware mogelijk maken. Ons nieuw voorgestelde algoritme toonde verbeterde prestatieschaling bij extra planningsbudget, niet alleen in data-efficiëntie maar ook in rekentijd-efficiëntie in vergelijking met eerder werk.

Het laatste Hoofdstuk 4 benut onze vinding dat sequentiële Monte-Carlo gebaseerde planning de rekentijdcomplexiteit kan verbeteren door computationeel budget te verschuiven van het leren van neurale netwerken naar extra planningsbudget voor het vergaren van hogere kwaliteit data. Aangezien we in meta-learning settings doorgaans te maken hebben met leren van trajectgebaseerde data, zien we vaak dat wanneer trajecten langer worden, updates van een neurale netwerkarchitectuur duurder worden. Dit komt doordat de computationele kosten van afgeleides bepalen voor het model toeneemt met trajectlengte, wat knelpunten in schaling veroorzaakt. Updates kunnen bijvoorbeeld de geheugenlimieten van de hardware overschrijden, waardoor kleinere batches sequentieel verwerkt moeten worden, wat de doorvoer vermindert. We constateren dus opnieuw dat de prestatieschaling bij extra planningsrekenkracht in deep meta-reinforcement learning aanzienlijk verbeterd kan worden wanneer uitgerust met onze ontworpen sequentiële Monte-Carlo planner. In tegenstelling tot voorgaande algoritmes houdt onze planner expliciet rekening met hoe onzekerheid evolueert in het licht van nieuwe data. Als gevolg hiervan richt onze methode zich direct op een optimaal beleid dat exploratie en exploitatie afweegt, zonder *ad-hoc* exploratieheuristieken.

Samenvattend draagt dit proefschrift bij aan de stand van probabilistisch reinforcement learning voor onzekerheidsgebaseerde besluitvorming. We laten zien dat Bayesiaanse optimalisatie en deep reinforcement learning zeer vergelijkbaar geformuleerd kunnen worden vanuit een probabilistisch oogpunt, en dat slechts (relatief) kleine aannames en keuzes van benaderingsgereedschappen ons van de ene setting naar de andere verschuiven. Dit is een bijzonder nuttig inzicht voor onze toepassing van autonome laboratoria, aangezien op het moment van schrijven Bayesiaanse optimalisatie een meer gevestigde aanpak is in de praktijk. Onze bevindingen vestigen een sterkere gemeenschappelijke basis tussen de ogenschijnlijk gescheiden velden van Bayesiaanse optimalisatie en reinforcement learning, en bieden een principiële grondslag voor het overdragen van benaderingsgereedschappen tussen deze gemeenschappen.

Acronyms

AI	artificial intelligence
BBO	black box optimization
BO	Bayesian optimization
DOE	design of experiments
DP	dynamic programming
EM	expectation-maximization
GPU	Graphical Processing Unit
MCTS	Monte-Carlo tree search
MDP	Markov decision process
NP	nondeterministic polynomial time
OOD	out-of-distribution
RL	reinforcement learning
RQ	research question
RV	random variable
SGD	stochastic gradient-descent
SMC	sequential Monte-Carlo

1

Introduction

Sometimes science is more art than science. A lot of people don't get that, Morty.

– Richard D. Sanchez, in *Rick Potion #9* (2014)

Automation of general industrial or scientific processes holds immense potential for societal advancement. Its application can accelerate innovations and discoveries, and shorten the time required for new technology to reach global markets. For instance, in the pharmaceutical industry, autonomous experimentation can reduce development time for new treatments, while in manufacturing, automation can increase factory throughput. It also contributes to improving quality of life by reducing workloads for employees. Moreover, automation can enhance environmental sustainability by optimizing the use of available resources.

An essential part in realizing automation is knowledge gathering. To improve an existing methodology a designer must have sufficient understanding of the subject to propose meaningful changes. Such an understanding is built through observations (evidence) combined with sound reasoning and skepticism, which underpins the *scientific method*. However, this process of carefully acquiring and internalizing knowledge also forms a bottleneck for applications of interest. It can take humans decades or generations to form a reliable comprehension of particular subjects or observed phenomena. This delays and limits the number of meaningful contributions that can be made. This has stirred ambitions to automate the process of science itself (Waltz & Buchanan, 2009), ranging from narrow tasks that streamline measurement-taking to reduce human error and increase throughput, to tackling the full scientific pipeline (Lu et al., 2024).

To realize 'automated science', recent advancements in **artificial intelligence (AI)** and robotics are proving to be indispensable. Modern **AI** methods can aid in decision-making by accounting for vast amounts of historical observations (data; Halevy et al., 2009), while robotics enables these decisions to be actualized in the physical world. These approaches have been successfully demonstrated in diverse domains, such as playing simulated games

1

Example 1.1: *In Vitro*; Automated Laboratories

An autonomous laboratory, also known as a self-driving laboratory (Häse et al., 2019), integrates robotics and artificial intelligence to form a complete scientific pipeline for the discovery of new materials, pharmaceuticals, and biotechnologies. Figure 1.1 illustrates an example of a robotic arm used to precisely assist in powder dosing (Merchant et al., 2023).



Figure 1.1: **The A-Lab at University of California, Berkeley.** Guided by a learning algorithm, and as part of a broader experimentation process, this robot arm can grab, open, and dispense materials within its reach. This creates a streamlined process for precise and reliable powder dosing. Photo credit: Marilyn Sargent/Berkeley Lab (licensed under CC BY 4.0 via Springer Nature).

In this example, a machine learning model that leverages a large database of example materials (Jain et al., 2013) predicts the stability of novel material combinations. An optimization algorithm then uses such predictions to find the most informative material combinations to test in practice, both to validate the model and synthesize many new materials. This robotic automation of experiments creates a high-throughput closed-loop system, which reduces the actual user's role to just evaluating the synthesized products.

(V. Mnih et al., 2015, Silver et al., 2016), protein folding prediction (Jumper et al., 2021), generation of video (Blattmann et al., 2023, Dosovitskiy et al., 2021), algorithmic discovery (Fawzi et al., 2022b, Mankowitz et al., 2023b), drone navigation (Kaufmann et al., 2023), or for scientific discovery in automated laboratories (see Example 1.1; Abolhasani and Kumacheva, 2023, Merchant et al., 2023).

Beyond integrating large amounts of data, a defining characteristic of **AI**-based methods is their ability to adopt abstractions that generalize across problem domains. This means that methodological advancements made in learning to play a videogame can lead to improved decision making in a separate field (Fawzi et al., 2022b, Jumper et al., 2021, Kajita et al., 2020, Mankowitz et al., 2023b, Segler et al., 2018, Shi & Evans, 2023). Nonetheless, despite this promised generality, different problem domains often necessitate domain-specific algorithmic features. For instance, algorithms that learn to control and navigate drones (Kaufmann et al., 2023) require fast response times and must process high-dimensional video data. In contrast, automated scientific experimentation (Merchant et al., 2023) allows for much longer response times. However, an algorithm now needs to handle more diverse input modalities like text, ordinal, and image data. At a fundamental level, these two applications share similar abstractions, which enables algorithm reuse. However, their distinct characteristics necessitate adaptations and modifications to the algorithms for each domain As a result, algorithms that were designed across separate fields often incorporate different insights or assumptions appropriate to their domain.

We predict that the convergence of solutions from different fields will yield further algorithmic improvements. As an example, this is how transformer models from language processing were used for computer vision (Dosovitskiy et al., 2021) with great success. Inspired by this, this dissertation aims to 'advance general sequential decision-making algorithms for scientific experimentation.' We pursue this ambition through the intersection of two dominant algorithmic approaches in **AI** that attempt optimal sequential decision making under uncertainty. These two approaches being **Bayesian optimization (BO)** (Mockus, 1989) and **reinforcement learning (RL)** (Sutton & Barto, 2018).

To provide context on how we aim to achieve this, this chapter first motivates an abstracted model that captures the complexity of real-world scientific experimentation. This abstraction builds up from the historically relevant framework of optimal experiment design, to numerical optimization for which **BO** is popular, and finally to the agent-environment framework relevant to **RL**. We then describe how Bayesian search methods enable general algorithmic solutions for the agent-environment framework, but faces challenges in practical scaling. From this, we define a scope of challenges for this dissertation from which we motivate our research goals and position our contributions.

1.1 Abstracting Sequential Experiment Design

At the foundation of science is the accumulation, introspection, and utilization of knowledge (Okasha, 2016). Specifically, science deals with *epistemology* (Audi, 2010) by formulating hypotheses about the real world which can be validated through empirical evidence. In other words, through observations made from experiments (Peirce, 1883). However, science is also *pragmatic* in the sense that new knowledge should serve societal or individual needs (Peirce, 1878). Thus, when handling controllable processes in practice that rely strongly on measurement, a natural question to ask is: how does one design the best possible experiment and what defines 'best'? This question compels the concept of optimal experiment design.

However, real-world scientific experimentation is not limited to just formulating optimal experiments. We need to account for how knowledge from experiments is used downstream in order to focus the questions we want to ask. Furthermore, experiment designs need to consider practicalities, such as (but not limited to): the feasibility of certain measurements, noise in the observations, budget and time constraints relative to the size of the problem, possible confounding, or early stopping criteria. Although we can develop experiment strategies for every such characteristic, it is more useful (in the long-term) to create abstractions of the underlying problem that enable more general solutions. Therefore, this section builds towards an abstraction for optimal experiment design such that we can capture the complexity of real world scientific experimentation.

1.1.1 Optimal Experiment Design

Classical optimal experiment design describes estimation of the generative parameters of an observed (and controllable) phenomenon that is optimal under some statistical criterion given an assumed mathematical model of the real world (Peirce, 1876, K. Smith, 1918). With "generative parameters" we mean the true, but unknown, input values of the observed process, such that, if we knew their values we could use our model to perfectly

reproduce the observations aside from independent noise. We refer to such a mathematical model conditional on (optimal) estimates or distributions of its generative parameters as a *statistical model* (Pearl, 1988). Most importantly, an optimal design guarantees unbiased and minimal variance estimation of statistical models at a tolerated level of precision (Kiefer, 1959). The desired precision then controls the costs or number of experiments that we need to perform. To put this into context, when we use statistical models for evaluating counterfactuals or to make predictions about future events, their reliability depends on a practical trade-off between robustness and efficiency.

To design an optimal experiment in practice, perhaps the most well known principles to follow are those described by Fisher (1935), also known as the **design of experiments (DOE)**. It formulates how to reliably perform inference or make statistical claims when comparing reference measurements (also known as baseline or control) to experimental measurements. Most importantly, in order to achieve unbiased and efficient estimation of statistical models, the **DOE** advocates that controllable factors of experiments should be exhaustively varied to reduce confounding, through randomization or blocking, and that experiments must be replicated to reduce uncertainty.

Unfortunately, when the number of controllable factors in an experiment design becomes relatively large, typical **DOE** becomes impractical or economically prohibitive. The reason being that the dimensionality over possible combinations of the experiment factors grows exponentially. As a consequence, estimating our statistical models at a certain level of precision, in the worst case, will also require exponentially more experiments. This is an effect of *the curse of dimensionality* (Bellman, 1957). By making mild assumptions on the underlying problem, however, it is possible to reduce this effect. For instance, fractional factorial designs assume that only a few factors have meaningful effects on the observations and that interactions between these factors are limited (Finney, 1943). This reduces the number of required experiments by a few factors in the exponent. Nevertheless, this is only an improvement, and the curse of dimensionality will always severely restrict the practicality of experimental setups.

1.1.2 Sequential Design as Numerical Optimization

The principles of optimal design prescribe how to reliably setup a single experiment, extract information from measurements, and draw sound conclusions. In practice, however, we typically perform small experiment designs that build on each other incrementally (Wald, 1945) rather than through formulating one optimal experiment design. Although such a sequential design induces delays in feedback, it also reduces the effect of the curse of dimensionality by spreading measurements out over time and enabling early-stopping of experiments (Shiryaev, 1978). This means that experiments can be terminated prematurely when sufficient information has been gathered for downstream use of the statistical model, such as answering a statistical hypothesis at an accepted level of confidence.

This integration of the downstream use and estimation of our statistical models reveals a practical trade-off. We can reduce the precision of statistical models that would be estimated through a classical optimal design such that they are sufficiently accurate and feasible to obtain for practical use. However, the objective of experiment informativeness often competes with practical feasibility, and both are challenging to formulate accurately.

Thus, to effectively handle this trade-off, it is useful to adopt a *numerical optimization* (Nocedal & Wright, 2006) abstraction for sequential design.

We rephrase sequential experimentation more precisely as a derivative-free optimization problem (Conn et al., 2009) where an algorithm sequentially proposes sample experiments X_t at every experiment step $t = 1, 2, \ldots, T$. Each sample experiment generates observations Y_t through an observed process f_t, which are then summarized through an *objective* h_t into a utility-value U_t for optimization:

$$\underset{\text{Experiment}}{X_t} \mapsto \underset{\text{Empirical Process}}{f_t(X_t)} \mapsto \underset{\text{Measurements}}{Y_t} \mapsto \underset{\text{Objective}}{h_t(Y_t)} \mapsto \underset{\text{Utility}}{U_t}$$

Thus, the optimizer must select sample experiments X_t that maximize the intermediate utility values U_t where information about this value is only revealed through function evaluations. This has the benefit of allowing for weak assumptions on f_t and h_t for the sake of generality. When the utility U_t, its accumulated value $U_1 + U_2 + \cdots + U_t$, or some other criterion is deemed sufficient, we can halt the optimizer and conclude the experiments.

To contrast this abstraction, a classical optimal design selects all sample experiments $X_1, X_2, \ldots, X_T$ simultaneously through a choice of h_t that captures statistical informativeness. Sequential design (Wald, 1945) adopts a similar choice for the objective h_t, however, the ordering and choice of experiments $X_1, X_2, \ldots, X_T$ is determined at each step t. This has the merit of enabling early-stopping if an ideal ordering can be found, which saves up time and resources. However, the ordering itself usually depends on a downstream application of the estimated model. Thus, the optimization abstraction generalizes the objective h_t to a broader class of functions (i.e., beyond statistical criteria) that better integrates the practical needs of our estimated model for selecting experiments.

Example Solutions

The utility of the 'derivative-free optimization' abstraction lies in enabling efficient and general solutions for diverse characteristics or assumptions on the process f_t and objective function h_t. For instance, evolutionary algorithms (Eiben & Smith, 2015) excel in problem settings where both f_t and h_t are cheap to evaluate. In other words, settings where the cost of designing an optimal experiment far exceeds the cost of sampling—prioritizing taking more, but less informative, sample experiments. Some example applications of this include automated antenna design (Hornby et al., 2006) or neural network structure learning (Stanley & Miikkulainen, 2002). In other settings where evaluations of f_t and h_t can be parallelized or distributed to a large extent, successful solutions even include advanced forms of random search (Jamieson & Talwalkar, 2016, Li et al., 2018). This applies in (simulated) parameter search problems like configuring machine learning algorithms (Bergstra & Bengio, 2012). Conversely, when sampling from f_t is expensive or slow, memory-based algorithms like **Bayesian optimization** (Mockus, 1989) are popular. These methods spend more time and resources on formulating (computing) experiments given currently available information, and have successfully been applied in automated laboratories (Roch et al., 2020), tuning neural network hyperparameters (Hutter et al., 2019, Silver et al., 2016), or configuring particle and quantum physics simulations (Carleo et al., 2019, Ilten et al., 2017).

1

Example 1.2: *In Silico*; Asynchronous Parameter Optimization

Numerical optimization algorithms that leverage distributed evaluations can achieve much better time-efficiency compared to fully sequential algorithms by trading-off the additional cost sampling more points. Although these algorithms increase throughput, it complicates the problem of strict sequential experimentation to general dynamic control.

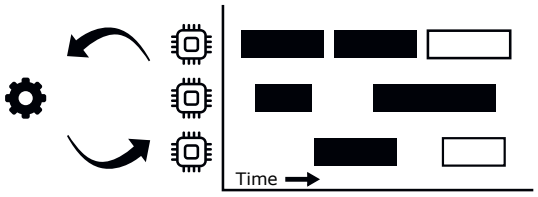


Figure 1.2: **Interaction of a configuration selection algorithm (gear; left) with a compute cluster (right).** The algorithm requests jobs for experiments, which are then allocated to timeslots for execution. These jobs can return feedback intermittently and upon completion.

A common example of this is in simulated parameter optimization, as shown in Figure 1.2. In this context, a configuration selection algorithm can choose to ignore or take pending experiments into account when proposing new experiments. Moreover, under strict time constraints, the algorithm can even cancel ongoing experiments when intermediate results provide evidence on better configurations to test. The optimal decision to take generally entails solving an intractable planning problem (Bellman, 1957).

1.1.3 An Agent-Based Abstraction for Dynamic Control

In sequential experiment design we perform (small) consecutive experiments to extend and build on accumulated knowledge to answer practical questions. Although the numerical optimization abstraction is holistic in this sense, it still fails to capture more complex settings of scientific experimentation. Of course, this doesn't prevent derivative-free optimization algorithms from being useful for these problems, as hinted through by the diverse set of examples from the previous subsection. However, it does introduce a degree of *model misspecification* (D. R. Cox, 2006), which can have varying degrees of (possibly negative) impact depending on the severity.

Arguably, the most pressing unaddressed assumption is that the conditions between sequential measurements are static. This states that there can be no confounding if we assume that the empirical process and our objective function from the previous subsection $Y_t = f_t(X_t)$ and $U_t = h_t(Y_t)$ are time invariant (i.e., $f_t = f_{t+1}$ for all t). Although this is a useful simplifying assumption on the methodology side, it is practically unrealistic aside from the most extremely controlled experiment environments.[1] For instance, certain experiments for X_t can break or wear down measurement equipment, leading to a difference in observations when reproducing experiments, $f_t(X_t) \neq f_{t+1}(X_t)$. This does not always depend on the experiments X_t themselves, but can also be induced by seasonal influences, the time of day, ambient temperature, or more. Moreover, changes in the costs of experiments, as given by the objective h_t, can fluctuate over time due to market dynamics.

[1] Even in the most extremely controlled laboratory environments it is technically impossible to satisfy $f_t = f_{t+1}$ in a strict sense. It is only ever realistic to make regularity assumptions between the two.

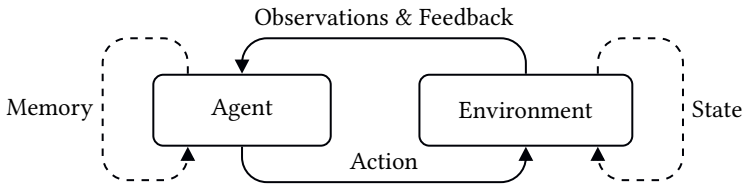


Figure 1.3: **The agent-environment model to abstract general closed-loop control.** The agent receives observations (feedback) from the environment based on the actions it proposes, and each action changes the environment's unrevealed state. To optimize the feedback, the agent should maintain an internal representation (memory) of the environment's state through these observations to determine the ideal actions.

Another restricting assumption is the closed-world model that consists only of strict sequential experimentation and feedback. For instance, consider the scenario sketched in Example 1.2 where a parameter configuration algorithm (Hutter et al., 2019) can perform many simultaneous (simulated) experiments and where the experiments can also yield intermediate feedback. Suppose that we want to trade-off shorter experiment times at the cost of potentially performing wasteful experiments. Although a 'numerical optimization' abstraction can integrate this property into an objective h_t, typical algorithms cannot directly deal with asynchronous feedback on the experiments as they can only propose new experiments. Instead, a better solution could integrate intermediate data and terminate long running experiments if said data reveals better experiments.

The Agent-Environment Model

To enable general solutions that addresses the aforementioned complexities, we require further abstraction of sequential experimentation to account for non-stationarity and dynamic control aspects within experiments. This can be effectively modeled using an agent-environment framework, as depicted in Figure 1.3. This model is common in **reinforcement learning** (Sutton & Barto, 2018) and is also referred to as general closed-loop or feedback control (Bertsekas, 2019, Powell, 2021). Specifically in the context of sequential experiment design, this more general formulation is analogous to the *identification for control* problem for non-linear dynamical systems (Nelles, 2001); we direct the reader to the work by Gevers (2005) for a complementary discussion to our abstraction.

In the proposed model, rather than just selecting experiments, an agent (our algorithm) now dynamically selects actions that indirectly *lead* to experiments—like the robot arm of Example 1.1 that can move its joints to select and dither materials to test new material combinations. Indeed, the "sequential" experimentation loop from the previous subsection still exists, but it is now embedded within a more complex closed-world system. Each action by the agent can update an unobserved state of the system which is only partially revealed through observations and feedback. On top of experiment measurements and utility values, this data can now include much broader information like the energy expenditure of a robot arm or the current number, costs, and runtime of experiments. To simultaneously account for uncertainty in this dynamic process and select actions that lead to the best experiments, an *optimal agent* must generally maintain an internal representation of the environment to adapt its experimentation strategy based on all previous data (Duff, 2002, Spaan, 2012).

1

Summary

In this section we introduced how optimal experiment design aims to uncover a statistical model of the real world in a way that guarantees a level of statistical precision. Despite existing frameworks for principled experiment design, like **DOE**, optimal design is often infeasible due to the *curse of dimensionality*. It is possible to cope with this problem through sequential experiment designs, as these can spread large designs out over time and prematurely terminate experimentation if the statistical model is deemed sufficiently accurate. This criteria reveals a trade-off between practicality of experiment design and the obtained model precision; which motivated abstracting experimentation into a more general optimization loop. Despite this abstraction having lead to many practically successful algorithms, we argued that sequential interaction with a static empirical process still assumes a misspecified model for real world experimentation. This lead to the final agent-environment framework, which can deal with non-stationarity in the experiment loop and general dynamic control. We will henceforth use this final framework to formulate our solutions for tackling the problem of scientific experimentation in the real world.

1.2 Scope: Challenges in Scaling Bayesian Search

The agent-environment model of Figure 1.3 is a thoroughly studied framework within **RL** and **dynamic programming (DP)**, and has prompted diverse algorithmic solutions. The two central themes in such solutions are 1) how to effectively handle uncertainty and 2) how to perform optimal decision making. Arguably, the most common and straightforward approach to reason about uncertainty is through probability theory, specifically through a Bayesian lens (Jaynes, 2003). Combining the Bayesian framework for handling uncertainty with optimal decision making motivates the framework of *Bayesian search*.

Bayesian strategies represent uncertainty for closed-world systems involving **random variables (RVs)**, like our agent-environment model, by defining a hypothesis set over these **RVs** and coupling individual hypotheses with (infinitesimal) probabilities that capture a belief of plausibility. This *prior* belief can be updated over time as new information is revealed (R. T. Cox, 1946, Jaynes, 2003), we typically refer to the updated belief as the *posterior*. An ideal solution to the closed-loop control problem under uncertainty can be formulated as an agent that solves the planning problem (i.e., maximize expected returns) while simultaneously accounting for how its posterior belief will evolve (Duff, 2002). Concretely, this means that actions are selected not just by their expected immediate return under the current belief, but also by how they influence future belief updates, which in turn affect the returns.

This ideal also clarifies how the two methodologies central to this dissertation relate. Sequential **BO**, as introduced in the previous section, maintains exactly such a posterior belief over an unknown process, but typically selects experiments through (myopic) heuristic acquisition functions rather than planning over future belief updates. Deep **RL**, in contrast, explicitly optimizes decisions over long horizons, but rarely maintains an explicit belief over the environment. Both methodologies thus approximate the same ideal from different directions, and the contributions in this dissertation target these shared foundations, rather than the full scientific-discovery pipeline, such that improvements transfer to both.

Although conceptually straightforward, Bayesian search generally poses considerable engineering challenges. Firstly, the complete specification of **RVs**, their interactions, and suitable prior beliefs, is often far from trivial, and in many cases intractable. Second, given a prior belief and sample observations, we must be able to update the posterior distribution efficiently. Unless we impose stringent assumptions, exact posterior inference is generally intractable (**NP**-hard; Johnson, 1990) as it entails solving for many (nested) integrals (Cooper, 1990). On top of this, the planning problem is itself computationally demanding (Papadimitriou & Tsitsiklis, 1987, Puterman, 1994), as it must recursively incorporate Bayesian belief updates while operating over a exponentially growing decision space with respect to the number of variables (curse of dimensionality). Exhaustive Bayesian search therefore quickly gets infeasible in large domains, motivating the need for algorithms that are both flexible in their model specification and able to scale efficiently.

Approximate Inference with Deep Learning. At the time of writing, the most successful approaches for scaling up posterior inference and planning are those that use function approximation through deep neural networks (Goodfellow et al., 2016). Specifically, function approximation can help deal with the intractable integrals that arise in Bayesian inference through *amortization* (Gershman & Goodman, 2014). Amortization means that we perform an expensive fitting (or training/learning) phase such that computation needed for (sub-)problems can be compressed within a much cheaper function. The idea is that the fitted function can reuse stored results later when faced with similar problem settings and interpolate between solutions for unobserved problems through pattern recognition (Bishop, 2007). In this regard, deep neural networks are some of the most flexible methods that also allow efficient training.

A deep neural network defines a massively overparametrized statistical model that can functionally relate diverse input modalities like text, video, categorical data, and more. This functional relation defines an (almost) overencompassing guess to our aforementioned closed-world formulation of the real world while also allowing relatively uninformative specification of prior beliefs. However, this side-steps, rather than solves, the aforementioned formulation challenge by trading principled model specification for generality. Imposing uninformative priors in such a way demands a fitted function to find or rule-out underlying relations and patterns strictly within its parameter-space given data (Frankle & Carbin, 2019, Halevy et al., 2009, Malach et al., 2020). Thus far, much of machine learning research and engineering has focused on designing neural network functions (architectures) in a way that 1) efficiently enables such parameter search and 2) reasonably improves in performance when increasing model size and data (Kaplan et al., 2020). It is exactly the philosophy of leveraging computational resources and datasets at unprecedented scale that has enabled considerable scientific and societal breakthroughs, as exemplified by algorithms like AlphaFold (Jumper et al., 2021) for protein structure prediction and by content generative models like ChatGPT (OpenAI et al., 2024) for text or Stable Diffusion (Rombach et al., 2022) for images.

So far, it is seemingly sketched that amortization through deep learning also enables the scaling of Bayesian search to large problem domains. Integrating such approaches within an interactive agent-environment loop, however, leads to unique challenges in algorithm design. Leveraging non-linear models for closed-loop control systems generally

1

leads to unpredictable long-term behavior (Strogatz, 2024). The choices for neural network architecture and parameter search, on top of approximation errors and (approximate) planning for optimal sequential decision making, only exacerbate this unpredictability due to inherent complexity. In other words, it is unclear what assumptions and patterns agents make, reuse, or interpolate between during deployment—or whether their solutions are even 'correct' at all (Jacot et al., 2018, Radhakrishnan et al., 2024). This complicates or downright prevents us from developing algorithms that guarantee behavioral properties like robustness (Morimoto & Doya, 2005), safety (Chow et al., 2018), alignment (Amodei et al., 2016, Langosco et al., 2022, Pan et al., 2021), and performance (Dulac-Arnold et al., 2021, Kakade, 2003). The remainder of this section, thus, defines a scope of current challenges that we argue are limiting effective and responsible deployment of this technology.

1.2.1 Semi-Amortization for Generalization

The point of amortization through function approximation methods is to cast expensive inference procedures to a much cheaper pattern recognition task (Bishop, 2007). Given training data, fitted functions can reuse patterns to new (unseen) inputs instead of exhaustively recalculating solutions. However, data is always limited in practice. This effectively guarantees that agents will encounter scenarios where training is imbalanced, mismatched, or does not cover many edge-cases with respect to downstream tasks—see Example 1.3 of how this can quickly arise in practice. Yet, we still want agents that use possibly mismatched amortization to perform well, or at least not break, in such scenarios.

This specific problem is also referred to as **out-of-distribution (OOD)** generalization (Yang et al., 2024), which itself is an instance of the classical bias-variance tradeoff in statistical learning theory (Hastie et al., 2009, Luxburg & Schölkopf, 2011). As hinted at earlier, **OOD** generalization is particularly difficult with deep neural networks since the complexity of their functions prevent or obstruct insight. What patterns and rules neural networks extract during training is currently not well understood (Jacot et al., 2018, Radhakrishnan et al., 2024) despite defining the downstream behavior of agents. Do learned patterns represent a fundamental rule of the real world? Or do they memorize information for accurate predictions on the training data, and only happen to perform well in limited testing scenarios? Moreover, even accurately formulating generalization on downstream tasks is difficult since in dynamic control settings (Kirk et al., 2023) and high dimensional spaces (Balestriero et al., 2021) many scenarios can be contrived such that test cases always lie outside the training data. To put it briefly, there are still many open problems within **OOD** generalization that underpin active topics of research.

To improve **OOD** generalization for agent-environment based control, we restrict our focus to *semi-amortization* methods for which we emphasize two distinct but related topics. Semi-amortization refers to methods that can leverage additional compute resources and data during deployment to perform corrections to amortized solutions (Kim et al., 2018, Margossian & Blei, 2024). This principle was, for example, used by AlphaGo to surpass human champions in the game of Go (Silver et al., 2016, 2017). To obtain this, we firstly need agents to reliably recognize **OOD** scenarios, which requires efficient methods for uncertainty quantification on the representations learned by neural networks. Then secondly, we need methods that can correctly perform finetuning to adapt to unseen

Example 1.3: Out-Of-Distribution Model Input and Output

In machine learning it is common that statistical models need to predict on test data that is often distinct from the training data distribution. Despite this mismatch, the underlying rules of the data are typically assumed to be invariant. Unfortunately, it is infeasible to exhaustively cover all possible combinations of the data during training. Effectively, this means that agents will frequently encounter **out-of-distribution** scenarios at test-time.

Image input. Consider the task of solving a maze from images, where the training data contains irrelevant information such as background color (Langosco et al., 2022). An ideal agent should be invariant to this color so it only reuses solutions when the maze stays the same and not the color.



Text input. Similarly, consider two text instructions for an agent (van Leeuwen, 2023),

"Move to your left." $\iff$ "Move opposite to your right."

Both instructions should lead to the same optimal behavior despite being formulated differently. Assuming that the distinct concepts of 'opposite', 'left/right' and 'move' are within the training data, an ideal agent should generalize to new, unseen, formulations.

Constrained output. Finally, consider an agent controlling a robot arm with multiple joints. The designer suspected possible test-time defects in the joints. Thus, an agent was trained on scenarios where one action dimension would receive an inactivity constraint to reflect such a scenario. However, this only considers a per-joint basis for potential failures. If multiple joints simultaneously incur defects during test-time, an agent must generalize between solutions of the per-joint failures.

scenarios. Since deep neural networks are massively overparametrized functions, one challenge is to deal with the computational burden for computing accurate uncertainty measures on the architecture or perform updates for correction. Another challenge is that naive model correction often violates statistical assumptions of the data in dynamic control scenarios. Finally, we also desire methods that can readily switch between providing fast but wrong solutions and slower but more accurate solutions.

1.2.2 Scalable (Approximate) Bayesian Planning

Assuming that we have an effective way to deal with uncertainty quantification, the second key component of Bayesian search is to perform optimal decision making. Although this is conceptually rather straightforward, it typically entails solving a computationally demanding planning problem (Bellman, 1957). Briefly put, this is because the optimal action at any decision event depends on all actions that follow it, and evaluating all such future paths incurs the curse of dimensionality. In other words, this emphasizes a more general challenge on how to tractably perform Bayesian search. Although diverse methodologies exist to attempt tractable search, each with their distinct assumptions and challenges, we specifically restrict our focus on dealing with the challenges of **BO** and **RL**.

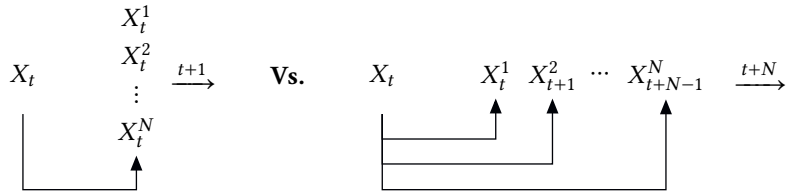
To this end, methods like **BO** (Mockus, 1989) simplify the planning problem by breaking dependencies on the future. In contrast, **RL** methods (Sutton & Barto, 2018) use amortization and sampling to iteratively update local solutions to the planning problem, eventually converging to the true (global) solution given enough samples and computation. When using deep neural networks for amortization this approach is also referred to as deep **RL**. Although **BO** methods harshly simplify the problem, this does often allow agents to satisfy the optimal Bayes decision rule within this model (Jaynes, 2003, Lehmann & Casella, 1998). In Bayesian **RL** methods this is considerably more difficult and expensive to obtain.

However, this discussion of assumptions becomes more nuanced when we link back to the motivating use-case of self-driving laboratories (Example 1.1). One common characteristic of this problem in practice is that of batched actions (González-Duque et al., 2024a), or somewhat similarly slate recommendation in **RL** (Deffayet et al., 2023). As illustrated in Example 1.4, this problem means that individual actions factorize in terms of multiple independent sub-actions, where the contribution of sub-actions to the overall optimization problem complexly depend on each other. A common strategy for filling in multiple individual items in such a batch is to model the problem sequentially and fill the items through **DP** (also known as liar strategies). However, this creates a scenario where concepts of **RL** and **BO** are mixed, since actions inside the batch are sequentially dependent, but actions outside the batch are not. Of course this is only one example, in practice multiple practical features may necessitate simplifications, generalizations, or additional structuring to obtain feasible algorithms. An essential question that then arises, is how to best tackle this question of model structuring in a sufficiently general manner.

So, the first challenge we identify is how to best contrast and evaluate assumptions between **BO** and **RL** methods. From this we can identify a second challenge as trying to leverage insights from this to improve both of these two existing methodologies. In turn, this creates multiple subchallenges in how to effectively deal with combined **RL** and **BO** related challenges. In similar spirit to the semi-amortization challenge introduce in Section 1.2.1, the main subchallenge we want to highlight is that of performing approximate planning that can scale well to increased computational resources. In other words, how do we design methods that can perform approximate Bayesian search and interpolate between solutions that are 1) fast but inaccurate or 2) slow but more accurate.

Example 1.4: Action Specification for Batched Experimentation

Instead of constructing one large experiment design, an agent can also generate smaller experiment designs sequentially (see Section 1.1.2). Anything between a full design and singular experiments, is also known as batched experimentation. For agent design, there are multiple ways to structure how such batches should be constructed.



As illustrated above, for some batch-experiment X_t, an agent can either generate an entire design instantaneously or generate sample experiments X_t^i one-by-one. Arguably, the sequential method makes fewer assumptions on the environment by spreading a potentially large batch out in time. The batch-size N becomes part of the environment dynamics and can vary dynamically; the agent can generalize to new domains without modification. The consequence, however, is that this induces delayed feedback for the first batch-items. In contrast, using a batched structure allows the agent to receive immediate feedback, but it requires additional assumptions on how to fill in said batch, especially with varying N. Depending on desired agent functionality, a tradeoff should be made here between generality and imposing structure to ensure solution feasibility.

1.3 Aim and Research Goals

This dissertation takes steps towards improving our understanding and implementation of algorithms for sequential decision-making based on the principles of Bayesian search. We adopt a general research vision, that reads as:

> *To develop better and more general algorithms for sequential decision making under uncertainty, we work within a unified framework for Bayesian optimization and deep reinforcement learning to better understand and reason about these methods, including the assumptions they make and their approximation quality.*

This is first and foremost an attempt to jointly discuss the design choices and approximation methods employed in **Bayesian optimization** and deep **reinforcement learning**, without having to visit each of these topics distinctly. As sketched in the previous section, we pose that **RL** tackles the sequential decision making problem in a more general (and possibly more performant), but more computationally involved manner. However, the relative simplicity of **BO** still makes it an attractive alternative with often competitive performance. Thus, our focus will predominantly be on deep **RL** for generality, however, we integrate the discussion of **BO** as a way to bridge these two approaches. Secondly, our outlined vision shapes our approach on designing, applying, and structuring general models. It

1

constrains us to leverage methods that reliably scale up in performance when given more computational resources, which can be easily transferred to differing domains.

To work towards the general aim of this project, within this encompassing vision, we break it up into manageable **research questions (RQs)** to address specific challenges, position it to existing approaches, and further focus the dissertation's scope. These **RQs** are listed below accompanied with a short isolated motivation. The first **RQ** focuses the general aim to two approaches of interest:

> **Research Question 1:** *Can we capture optimal sequential decision-making in a probabilistic model that generalizes Bayesian optimization and reinforcement learning?*
>
> When designing solutions to a given problem, it is often useful to adopt a standardized language to effectively reason about the appropriate tools or approaches to apply. A unified probabilistic formulation enables us to position Bayesian optimization and reinforcement learning within a more general class of methods.

BO and **RL** have been described from many viewpoints already in prior work (Mockus, 1989, Sutton & Barto, 2018), including probabilistic interpretations of optimal control (Kalman, 1960, Levine, 2018, Ziebart, 2010, Ziebart et al., 2010). However, they are usually discussed in isolation despite their strong similarities. Viewing both methodologies from a similar perspective allows us to more easily draw conclusions that have consequences to both fields, despite focusing on one singular branch of methods. For instance, recently deep reinforcement learning methods have been used for **black box optimization (BBO)** problems that outperform classical **BO** methods (Y. Chen et al., 2017). Although this sounds promising, because their approach is fully framed within **RL**, it does not explain why their results outperform **BO** in a setting where **BO** typically shines. So, to better understand this, we write our second **RQ**:

> **Research Question 2:** *How do contemporary deep reinforcement learning algorithms fit within our framework? What can we say about the quality of their approximations?*
>
> Practice shows that deep reinforcement learning algorithms are highly effective in tackling the planning (under uncertainty) problem in its full generality by using neural networks for approximate inference. However, they also often exhibit brittleness. Thus, our aim is to study how successful current deep reinforcement-learning algorithms fit within our framework and how well they perform their approximations. This can caution us when developing new algorithms.

Related to **RQ** 2, there has been prior work which demonstrates that reinforcement learning methods perform improper uncertainty quantification when training static agents to perform tasks involving inference (Beck et al., 2023, Xiong et al., 2021). More generally, this can be shown as an approximation gap in amortized functional inference that is no longer being corrected for (Cremer et al., 2018, Margossian & Blei, 2024). However, completely correcting model approximations amounts to full online inference, and despite being theoretically sound, comes at significant computational burden compared to fully amortized inference.

Ideally, we would have a mechanism that can interpolate between these two extremes. Furthermore, to reduce computational latency, we want to focus on approaches that scale particularly well to distributed computation. Specifically, this includes specializing algorithms to make better use of hardware acceleration like **Graphical Processing Units (GPUs)**. This principle allows us to very easily scale to additional computation through vectorized operations.

In summary, this motivates the next **RQ** which investigates methods for approximating our probabilistic model in a way that scales well in practice but also enables our algorithm to perform online corrections to a learned (amortized) statistical model:

> **Research Question 3:** *How do we efficiently scale approximate inference within our model formulation?*
>
> When making very general or loose assumptions on a complicated probabilistic graphical model, the designer often encounters numerous (nested) intractable integrals. In such cases, numerical approximation becomes effectively compulsory. Practice shows that amortizing the cost of integration by predicting their solution through overparameterized predictive models can be highly effective. Although, such predictive models usually enable cheap evaluation and flexible prior specification, they also induce additional approximation error due to restricted representational capacity or a poor fit—as a result of local optima or insufficient data. Therefore, our aim is to investigate methods that can leverage both amortization for fast initial solutions and online inference to correct for these fast, but wrong, initial solutions.

1.4 Outline

The dissertation is structured as follows, to investigate both **RQ** 1 and **RQ** 2, Chapter 2 shows how current meta-reinforcement learning algorithms (Beck et al., 2023) estimate a general underlying probabilistic model. However, from a strict statistical viewpoint, we show how this estimation can be poor in practice. The subsequent chapters leverage this insight to motivate better learning objectives and semi-amortized approximation methods. Targeting **RQ** 3, Chapter 3 investigates such a semi-amortized approach for approximate policy inference, this chapter proposes improvements to sequential Monte-Carlo planning algorithms for scaling to distributed compute and addresses. Finally, Chapter 4 completes **RQ** 1 & 3 by building the sequential Monte-Carlo planner inside a learned Bayesian filter to account for uncertainty. This final chapter presents the closest approximation to the general model we desired, as given by our general research aim and vision. The following discussion chapter the links our findings back to the research questions and provides recommendations for the design of future algorithms. At the end of this manuscript, we give an overview of repositories and academic publications in Appendix 'List of Publications'.

2

Posterior Uncertainty in Meta-Trained Agents

A: I heard you were extremely quick at maths, what is 37×12*?* *B: 47!*
A: That's wrong... *B: Yes, but it was fast!*

As explained in the Introduction, powerful function approximation enables approximate learning of intractable computations into cheaper functions. In deep reinforcement learning, this has proven to be an effective approach for training models in an offline setting that can perform well and adapt rapidly to new problems. However, mismatches between the training distribution and later deployed tasks, often called the 'simulation to real' gap, are a common cause of brittle behavior at test-time.

This chapter investigates this phenomenon and contributes a new insight that commonly used approaches in memory-based reinforcement learning methods learn an overconfident estimation of uncertainty inside their model. We present our contribution within the context of meta-reinforcement learning, so this chapter starts by discussing this setup and the connection to amortized inference. Our approach is then described in Section 2.4 where we introduce the 'Laplace variational recurrent neural network' architecture, a method that provides us with an efficient and architecture independent approximation for distributional statistics of recurrent agents. We perform a short empirical study on how our method interplays with representation learning and the different assumptions of the agent's graphical model.

Our findings confirm prior results that amortized statistical inference is not statistically consistent, moreover we find that the uncertainty of learned statistical estimators reduces despite not being consistent. This shows that end-to-end meta-learning cannot distill reliable learning algorithms in and off itself. Instead, care must be taken when designing meta-learning losses and architectures to align the agent's representation with intended behavior.

The contents of this chapter have been published in (de Vries, He, de Weerdt, & Spaan, 2025).

2.1 Introduction

Reinforcement Learning (RL) concerns itself with making optimal decisions from data (Sutton & Barto, 2018). This is typically achieved by letting an agent generate data in an environment and then optimizing a cost function of the agent's parameters given this data. In meta-RL, an agent is trained to optimize an expected cost over a prior distribution of environments (Beck et al., 2023, Finn et al., 2017). The idea is then that, given a trajectory of new data, an agent can infer latent environment parameters and successfully adapt its action policy online. This is known as zero-shot or few-shot adaptation or learning (Beck et al., 2023). In recent years, this paradigm has shown impressive results, for example, by the Capture the Flag agent (Jaderberg et al., 2019) or the Adaptive Agent (Bauer et al., 2023).

In any meta-RL algorithm, an accurate approximation of the latent parameter distribution given data, also known as the task posterior distribution, is useful to quantify the agent's uncertainty (Grant et al., 2018). Accurate quantification of uncertainty enables agents to detect distribution shifts (Daxberger et al., 2021) or guide exploration through novelty signals (Osband et al., 2013, Sekar et al., 2020). Importantly, on deployment, distribution shift detection is essential for timely human intervention or retraining. This ultimately improves the robustness and efficacy of our algorithms and allows us to more reliably inspect failure cases.

A common approach in meta-RL is to model the task posterior with point estimates, e.g., using the hidden state of recurrent neural networks (RNN) (Y. Chen et al., 2017). However, this prevents us from exploiting useful distributional statistics. Another downside of using point-estimates is the increased risk of overconfidence unless the true posterior is sharply peaked at that particular point. This would imply that there exists almost no uncertainty about the environment, which is typically a strong and unrealistic assumption. As a consequence, point-estimate based meta-RL has been known to overfit to its training distribution, leading to brittle downstream performance (Greenberg et al., 2023, Xiong et al., 2021).

Arguably, a better approach would be to explicitly parameterize some full distribution (e.g., a Gaussian) (Chung et al., 2015, Zintgraf et al., 2021). However, approximate Bayesian methods are slower to train and are still often outperformed by simple point-estimate methods in terms of expected returns (Greenberg et al., 2023) even though they model the posterior more accurately. This could be explained by the fact that non-Bayesian methods enjoy reduced sampling noise, easier numerical representation, and improved model capacity by not having to learn a complex posterior model (Goyal et al., 2017, Hafner et al., 2019).

To get benefits from Bayesian methods when using non-Bayesian models, we introduce the *Laplace variational recurrent neural network* (Laplace VRNN) which utilizes the Laplace approximation to extend RNN-based meta-reinforcement learning. Our method can perform uncertainty quantification for non-Bayesian meta-RL agents without modifying the model architecture or loss function, and without needing to retrain any parameters. In other words, the consequence of the Laplace approximation is that we can apply it at any point during model training. When applied after training, this is often referred to as a *post-hoc* posterior (Daxberger et al., 2021). This allows us to make use of deterministic pre-training schedules

and benefit from their aforementioned advantages while also enjoying the benefits of Bayesian methods. Although the Laplace approximation has already been explored in meta-RL (Finn et al., 2018, Grant et al., 2018), it has not been applied in memory-based methods (Beck et al., 2023, Duan et al., 2016) which is what we explore.

The Laplace approximation is a simple method that only requires the *curvature* of a distribution's log-likelihood at a local maximum (Daxberger et al., 2021). For a Gaussian mean-field assumption on our posterior model (Bishop, 2007), we only require the Jacobian matrix of the RNN output with respect to its hidden state. This gives us a Gaussian distribution for the task posterior distribution centered at the RNN hidden state with inverse covariance equal to the sum of Jacobian outer products. This is a comparatively cheap approximation compared to typical methods that apply the Laplace approximation to the much higher dimensional neural network parameters (Daxberger et al., 2021, Grant et al., 2018, Martens & Grosse, 2015).

We empirically validate that our method can *reliably estimate posterior statistics of our non-Bayesian baselines without degrading performance* on supervised and reinforcement learning domains. Similarly to Xiong et al. (2021), our results show that non-Bayesian meta-RL agents do not learn consistent estimators, however, we also find that the learned representations are overconfident. This could be seen by inspecting the *post-hoc* posterior provided by the Laplace approximation, which showed low entropy while not converging to a stable distribution. Furthermore, when comparing our method against variational inference baselines, we find that our Laplace method performs on par in terms of mean returns. Ultimately, this shows that the Laplace approximation can complement (or serve as an alternative to) variational inference methods for uncertainty quantification, since we do not *learn* our local uncertainty but estimate this based on the model's fitted parameters.

2.2 Related Work

Meta-Reinforcement Learning Meta-learning has been described from various viewpoints, ranging from contexts (Hallak et al., 2015), latent-variable models (Garnelo et al., 2018, Gordon et al., 2019, Wu et al., 2020), amortized inference (Gershman & Goodman, 2014), and "learning to learn" (Beck et al., 2023, Hospedales et al., 2021, J. X. Wang et al., 2017). Applying these ideas to reinforcement learning has been gaining traction within the field recently, for example, learning maximum likelihood estimation algorithms (like our work) (Andrychowicz et al., 2016, Garnelo et al., 2018), probability density functions (Bechtle et al., 2021, Lu et al., 2022), or model exploration strategies (Gupta et al., 2018).

Related to our work is the neural process by Garnelo et al. (2018) which formalizes using meta-learning to infer a set of (global) latent variables for a generative distribution. Since we test on reinforcement learning problems, one could view our model as a type of non-stationary stochastic process or sequential neural process (Øksendal, 2003, Singh et al., 2019). This makes our model and optimization objective similar to the PlaNet model (Hafner et al., 2019), however, we do not condition our recurrent model on samples from the task posterior so we can obtain an analytical solution. This choice does reduce the non-linearity of our model, the topic of linear vs. non-linear state space models is still active research (A. Gu & Dao, 2024).

Bayesian Reinforcement Learning Learning an optimal control policy conditional on a task-posterior amounts to approximating the Bayes-adaptive optimal policy (Duff, 2002). In this framework an agent is conditioned on its current state and a history of observations, the history can then be used to produce a belief distribution over latent variables. The Bayes-adaptive optimal policy maximizes the environment returns in expectation over this belief (Duff, 2002, Ghavamzadeh et al., 2015, Mikulik et al., 2020, Zintgraf et al., 2021). Although meta-learning induces uncertainty only over the reward and transition function, many have also successfully tackled the problem as a general partially observable Markov decision process (Bauer et al., 2023, Y. Chen et al., 2017). Doing this is orthogonal to our method, however, we focus on the Bayes-adaptive framework.

Laplace Approximation The Laplace approximation has been explored for meta-learning in, for example, the model agnostic meta-learning algorithm (Finn et al., 2017, 2018) to achieve more accurate inference, or for continual learning (Kirkpatrick et al., 2017) as a regularizer for weight-updates. The main obstacle to using the Laplace approximation in practice is the computation of the inverse Hessian (MacKay, 1992), which alone has quadratic memory scaling in the number of model parameters. For this reason, in Bayesian neural networks, the block-diagonal factorization has become quite popular (Martens & Grosse, 2015), as used by TRPO (Schulman et al., 2015) or second order optimizers (Botev et al., 2017). Our method bypasses the costly Hessian problem by modeling a distribution on a small subset of the full parameter set. In contrast to doing Bayesian linear regression on the last layer of a neural network, our method can express multimodal distributions.

2.3 Preliminaries

We want to find an optimal policy π for a sequential decision-making problem, which we formalize as an episodic Markov decision process (Sutton & Barto, 2018). We define states $S \in \mathcal{S}$, actions $A \in \mathcal{A}$, and rewards $R \in \mathbb{R}$ as random variables that we sample in sequences. We write $H^i = \{S_t, A_t, R_t\}_{t=1}^T$ to abbreviate the joint random variable of episode $i \in \mathbb{N}$,

$$p(H^i) = \prod_{t=1}^{T} p(R_t|S_t, A_t)\pi(A_t|S_t)p(S_t|S_{t-1}, A_{t-1}),$$

where $p(S_1|A_0, S_0) \stackrel{\Delta}{=} p(S_1)$ is the initial state distribution, $\pi(A_t|S_t)$ is the policy, $p(S_{t+1}|S_t, A_t)$ is the transition model, and $p(R_t|S_t, A_t)$ is the reward model. To avoid confusion, we denote episodes in the *superscript* from $i = 1, \ldots, n$ and time in the *subscripts* from $t = 1, \ldots, T$. For convenience, we subsume the common discount factors $\gamma \in [0, 1]$ into the transition probabilities as a global termination probability of $p_{\text{term}} = 1 - \gamma$ (Levine, 2018) assuming that the MDP will end in an absorbing state with zero rewards. The objective is to find π^* such that $\mathbb{E}_{p(H)} \sum_t R_t$ is maximized.

2.3.1 Inference in Meta-RL

In contrast to single-task Reinforcement Learning (RL), in meta-RL we want to find the optimal policy π^* to a distribution over different environments. The agent typically does

not know which environment it is currently being deployed in and needs to adaptively switch strategies based on online feedback. We assume that the agent can adapt over *multiple* episodes $H^{1:n}$, as opposed to only one episode H^1, which is also known as zero-shot or few-shot adaptation (Beck et al., 2023). This approach can be formalized using the concept of global latent variables Z. For a fixed π, each trajectory H that we sample depends on a sampled latent variable $Z \sim p(Z)$, which could be interpreted as a unique identifier of the current environment. Our agent does not directly observe Z, but this variable influences the reward and transition models of the environment.

With full generality, we define the generative process,

$$p(H^{1:n}, Z^{1:n}) = \prod_{i=1}^{n} p(H^i|Z^i)p(Z^i|Z^{<i}, H^{<i}), \tag{2.1}$$

where the term $p(H^i|Z^i)$ indicates the sampling distribution of the environment under our current model for Z, and the term $p(Z^i|Z^{<i}, H^{<i})$ denotes the posterior distribution over latent variables Z given all the data we have observed so far. For brevity, we do not expand the sampling distribution $p(H^i|Z^i)$ here (see Appendix 2.A.1), however, this expression hides that the posterior model is also updated inter-episodically at every $S_t^i, A_t^i, R_t^i \in H_t^i$. Z^i. Note that this model is fully general, and is perhaps more common in continual-RL settings (Khetarpal et al., 2022), yet it reflects the model factorization and inference capabilities captured by most memory-based meta-RL methods (Duan et al., 2016). This generality can be both a feature and a downside; the agent can capture broad environment settings, but combined with function approximation it obfuscates what the agent learns and how posterior uncertainty is represented.

Amortized Inference The posterior $p(Z^i|Z^{<i}, H^{<i})$ from Eq. (2.1) is usually intractable; a common approach to deal with this is to use variational inference (Bishop, 2007). This approach represents the posterior with another distribution $q \in \mathcal{Q}$ within some simpler model class, and then chooses q to maximize a lower-bound to the data marginal (see Appendix 2.A.1), i.e.,

$$\ln p(H^{1:n}) \geq \max_{q \in \mathcal{Q}} \mathbb{E}_{q(Z^{1:n}|H^{1:n})} \sum_{i=1}^{n} \ln p(H^i|Z^i) - KL\left(q(Z^{\leq i}|H^{<i}) \| p(Z^{\leq i}|H^{<i})\right). \tag{2.2}$$

This involves a functional optimization problem to be repeatedly solved at every timestep. Therefore, a more desirable approach is to amortize this with a learned neural network f_θ that maps past observations and latents directly to a distribution $q \in \mathcal{Q}$ (Gershman & Goodman, 2014). In practice, this can be achieved by using a parametric family for q_ϕ and using f_θ to predict the parameters $\phi \in \Phi$.

To find the parameters θ that maximize the evidence lower-bound, we can derive an amortized learning objective. If we abbreviate the maximand from Eq. (2.2) as $\mathcal{L}(q, H^{1:n})$ and define $f_\theta : \mathcal{H}^n \to \Phi$ (omitting Z for brevity), we always have the inequality

$$\max_\theta \mathbb{E}_{p(H^{1:n})} \mathcal{L}(q_{\phi=f_\theta(H^{1:n})}, H^{1:n}) \leq \mathbb{E}_{p(H^{1:n})} [\max_{\phi \in \Phi} \mathcal{L}(q_\phi, H^{1:n})]. \tag{2.3}$$

This shows that **1)** with a sufficient function class for f_θ and optimal parameters θ^*, we obtain equality when $f_{\theta^*}(H^{1:n}) = \arg\max_{\phi\in\Phi} \mathcal{L}(q_\phi, H^{1:n}), \forall H^{1:n} \in \mathcal{H}^n$, and **2)** θ^* can be obtained through a straightforward training procedure. The l.h.s. requires us to be able to sample from $p(H^{1:n})$, and that $\mathcal{L}$ is end-to-end differentiable with respect to θ, which enables the use of stochastic gradient methods (Kingma & Ba, 2017).

From Inference to Control So far, we have mostly discussed how meta-RL agents can perform inference to latent variables of the environment for a fixed policy π. To define optimal behavior in the generative process we extend the probabilistic model of Eq. (2.1) using the control as inference framework (Attias, 2003, Levine, 2018, Toussaint & Storkey, 2006). This enables us to apply our Bayesian tools directly to our RL-agent in a theoretically sound manner and, for specific design choices, recovers the typical meta-RL training objective (Duan et al., 2016, J. X. Wang et al., 2017).

Control as inference reformulates classical RL as an inference problem by conditioning the distribution over trajectories $p(H^i)$ on a desired outcome $\mathcal{O}$. This outcome $\mathcal{O} \in \{0, 1\}$ is a binary variable indicating whether a trajectory achieves the outcome (1) or not (0). The likelihood of this outcome given a trajectory H^i follows the exponentiated sum of rewards, $p(\mathcal{O} = 1|H^i) \propto \exp(\sum_t R_t^i)$. Using Bayes rule, we can then infer the desired policy as $p(H^i|\mathcal{O} = 1)$.

Similarly to the latent variable posterior q_ϕ, we can estimate $p(H^{1:n}|\mathcal{O} = 1)$ through variational inference by defining a lower bound to the log-likelihood of the outcome variable $\ln p(\mathcal{O} = 1|H^{1:n})$ using a variational policy $q_\pi(A_t^i|S_t^i, Z_t^i)$. Given our choice for the outcome likelihood, this recovers a regularized RL objective (Geist et al., 2019) (see the derivation in Appendix 2.A.1),

$$\begin{aligned}\mathcal{L}(q_\phi, q_\pi) = \mathbb{E}_{q_\phi(H^{1:n}, Z^{1:n})} \sum_{i=1}^{n} \sum_{t=1}^{T_i} R_t^i - KL(q_\pi \| \pi) \\ - KL\left(q_\phi(Z_t^i|Z_{<t}^{\leq i}, H_{<t}^{\leq i}) \| p(Z_t^i|Z_{<t}^{\leq i}, H_{<t}^{\leq i})\right)\end{aligned} \tag{2.4}$$

which is an extension of the MPO objective by Abdolmaleki et al. (2018) to include the latent-variable posterior and its KL-term. The indexing of the conditional is slightly overloaded for brevity, it indicates that the task posterior is conditioned on all prior data.

Unfortunately, this lower-bound does not give a practical training objective for both inference or amortization as shown in Eq. (2.3), nor does it find the optimal policy given a fixed π. Therefore, practitioners often include the following design choices.

1. The true posterior $p(Z_t^i|Z_{<t}^{\leq i}, H_{<t}^{\leq i})$ is substituted by the previous variational posterior $q(Z_{t-1}^i|Z_{<t-1}^{\leq i}, H_{<t-1}^{\leq i})$, using only a "true" prior $p(Z_0)$ at time and episode 0 (Hafner et al., 2019).

2. The policy π is iteratively updated to the variational optimum q_π (Abdolmaleki et al., 2018).

3. KL-penalties are scaled by parameters β_π, β_q.

Finally, observe that the amortized objective for Eq. (2.4) (i.e., substituted into Eq. (2.3)) recovers the typical memory-based meta-RL objective when using a point-estimate (Dirac posterior) $q_\phi(Z|\dots) = \delta(\phi - Z)$ and ignoring the KL-penalties (Duan et al., 2016).

2.4 Laplace Variational Recurrent Networks

We introduce the Laplace variational recurrent neural network (Laplace VRNN) to make a relatively simple approximation to the variational task posterior $q_\theta \approx \hat{q}_\theta$, to be used in the lower-bounds of Eq. (2.2) and Eq. (2.4), using the Laplace approximation (Daxberger et al., 2021, MacKay, 1992). This enables the construction of proper distributions over the latent-variables without introducing additional variational parameters. The idea is that we use this to extend point-estimate methods (i.e., base RNNs) to use distributions at any point during training. For exposition, we introduce our approximation starting from a simpler variational distribution $q_\phi(Z_t|H_{<t})$, dropping superscripts. The full derivation is given in Appendix 2.A.2.

We use the predicted distribution parameters $\phi_t = f_\theta(H_{<t})$ as a helper variable, and interchange the notation $q_{\phi_t}(Z_t|H_{<t}) = q_\theta(Z_t|H_{<t}, \phi_t)$. Most importantly, the statistical amortization by Eq. (2.3) allows us to interpret the mapping $f_\theta : H \mapsto \phi$ as a learned summary statistic for the distribution of Z; i.e., a *maximum a posteriori* estimate. We assume that ϕ_t is computed autoregressively with a recurrent neural network (RNN) such that $\phi_{t+1} = f_\theta(S_t, A_t, R_t; \phi_t)$.

We then factorize using a mean-field assumption,

$$\begin{aligned} q_\theta(Z_t|H_{<t}, \phi_t) &= \frac{1}{q_\theta(Z_t|\phi_t)^{t-2}} \prod_{i=1}^{t-1} q_\theta(Z_t|S_i, R_i, A_i, \phi_t), \\ &= \exp[(2-t)\ln q_\theta(Z_t|\phi_t) + \sum_{i=1}^{t-1} \ln q_\theta(Z_t|S_i, R_i, A_i, \phi_t)] \\ &= \exp h_\theta(Z_t; H_{<t}, \phi_t), \end{aligned} \tag{2.5}$$

which gives us the target function h_θ that we wish to approximate (for the first step, see Lemma 2.1; Appendix 2.A.1). For a given θ and data $H_{<t}$, we use the second order Taylor expansion of $h_\theta \approx \hat{h}_\theta$ linearized at $\phi = \phi_t$, we then exponentiate $\hat{h}_\theta$ and renormalize. Assume that ϕ_t is *maximum a posteriori* to $q_\theta(Z_t|H_{<t}, \phi_t)$, then we obtain the Laplace approximation,

$$\begin{aligned} q_\theta(Z_t|H_{<t}) &\approx \frac{\exp \hat{h}_\theta(Z_t; H_{<t}, \phi_t)}{\int \exp \hat{h}_\theta(z; H_{<t}, \phi_t) dz} \\ &= \mathcal{N}(\phi_t, \nabla^2_\phi \ln q_\theta(Z_t|H_{<t}, \phi)|_{\phi=\phi_t}), \end{aligned} \tag{2.6}$$

where $\nabla^2_\phi \ln q_\theta$ is the Hessian of our log-posterior.

To complete our model from Eq. (2.2) and Eq. (2.4), we can solve the expectation over $\mathbb{E}_{q_\theta(Z_{<t}|H_{<t})}$ for the posterior $q_\theta(Z_t|H_{<t}, Z_{<t})$ at each time-step t, by assuming a convolution

of Gaussian densities. The result of this convolution is well-known to be another Gaussian with summed parameters (Bromiley, 2003),

$$\mu_t = \phi_t + \sum_{i=1}^{t-1} \mu_i, \tag{2.7}$$

$$\Lambda_t = -\nabla^2_\phi \ln q_\theta(Z|H_{<t}, \phi)|_{\phi=\phi_t} + \sum_{i=1}^{t-1} \Lambda_i. \tag{2.8}$$

In practice, we use a smaller window $H_{k:t-1}$ and $Z_{k:t-1}$ inside the conditional for efficiency. In summary, this implements an RNN where we sum the last k hidden states and covariances for the output-Gaussian, and where the covariances are produced by the Hessian of the log-posterior with respect to the hidden state.

The assumption that ϕ_t is *maximum a posteriori* is quite strict and assumes equality in the amortization objective Eq. (2.3). For a sub-optimal θ or insufficient function class f_θ, this can induce a first-order error in the Taylor expansion of the variational log-posterior, which results in a worse approximation. Furthermore, most RNN methods do not sum their hidden states for the predictive model, which mismatches our formulation. It is also not obvious what representations RNN-based meta-RL agents learn and, thus, which model factorization would best suit the agent for construction of our Laplace approximation. We investigate these technicalities and assumptions in the next section.

Special Case Our method obtains a particularly nice form if we choose $q_\theta(Z_t|S_i, A_i, R_i, \phi_t)$ to be standard Gaussian and use an uninformative prior for $q_\theta(Z_t|\phi_t)$. In that case, the Hessian of the log-posterior $q_\theta(Z_t|H_{<t}, \phi_t)$ becomes a sum of outer products of our RNN state Jacobians w.r.t. ϕ. Let $x_i = (S_i, A_i, R_i)$, this gives the inverse covariance,

$$\Lambda_t = \sum_{i=1}^{t-1} (\nabla_\phi f_\theta(x_i; \phi)|_{\phi=\phi_t})(\nabla_\phi f_\theta(x_i; \phi)|_{\phi=\phi_t})^\top, \tag{2.9}$$

which is cheap to compute with forward accumulation (Bradbury et al., 2018), see Prop. 2.3 in Appendix 2.A.2.

Posterior Predictive If we now choose a policy $\pi(A_t|S_t, Z_t)$ that is linear in Z_t, then our full model would recover a type of Gaussian process (Immer et al., 2021, Rasmussen & Williams, 2005). However, we model this term with another neural network $\pi_\psi(A_t|S_t, Z_t)$ to improve expressiveness. Our policy is then defined by the *posterior predictive* $\pi_\psi(A_t|S_t, H_{<t})$, which we compute using Monte-Carlo,

$$\int \pi_\psi(A_t|S_t, z_t) q_\theta(z_t|H_{<t}) dz_t \approx \frac{1}{k} \sum_{i=1}^{k} \pi_\psi(A_t|S_t, Z_t = z^{(i)}), \quad z^{(i)} \sim q_\theta(Z_t|H_{<t}), \tag{2.10}$$

overloading superscripts to index Monte-Carlo samples. This induces a finite mixture for the policy where $k = 1$ corresponds to posterior sampling (Osband et al., 2013). The

parameters θ and ψ were trained jointly in an end-to-end manner (as defined in the l.h.s. of Eq.(2.3)).

Interestingly, during training we found output aggregation to train more stably in the loss when $k > 1$. Thus, during training we chose to average the predicted logits of π_ψ over samples $z^{(i)}$, or in the continuous case, we averaged over the parameters of a parametric distribution (C. Wang et al., 2020), e.g., the mean and variance of a Gaussian (see Appendix 2.B.3 for a discussion).

2.5 Experimental Validation

In order to apply our method to memory-based meta-RL agents in a manner that didn't alter the network architecture, we required a few extra simplifications to our more general model from Section 2.4. Crucially, for the *non-stationary* assumption on the log-posterior only, this required us to omit the mean aggregation of Eq. (2.7). This wasn't needed for the stationary factorizations, see Appendix 2.A.2 for a more detailed discussion. To evaluate our factorizations and design choices, we performed experiments to answer the following:

1. **Utility:** Does our method give useful posterior statistics for a non-Bayesian baseline?
2. **Sensitivity:** What model assumptions for the Laplace VRNN are empirically effective?
3. **Performance:** When used as an alternative to variational inference, does the Laplace VRNN perform at least on par with existing methods?

Our point-estimate (RNN) baseline was implemented with a long-short term memory architecture (Hochreiter & Schmidhuber, 1997). The VRNN baseline (Chung et al., 2015) extends the RNN by predicting the mean and covariance for a Gaussian distribution as a transformation of the RNN output. For the RNN and VRNN we assumed a stationary factorization of the posterior $q_\theta(Z_t|H_{t-1})$, which is a simplification of the fully general posterior shown in Eq. (2.1) and most accurate to the true generative process. We intermittently created model snapshots of the point-estimate baseline (RNN) and finetuned these snapshots over our parameter grid for the Laplace VRNN and VRNN.

The purpose of the evaluation domains we chose, zero-shot regression, stochastic Bandit, and a sparse gridworld, is to have a variety of tasks that incrementally scale up in complexity but each contain a meta-learning aspect. This fits the framing of the amortization objective of Eq. (2.3): training over a distribution of tasks incentivizes the learned representation to internalize an amortized (Bayesian) inference procedure. Based on this representation assumption, we can give meaning to our Laplace VRNN, as it allows us to extract, and inspect, what kind of posterior the representation has actually learned.

All experiments were repeated over $r = 30$ seeds (number of network initializations), we tested intermediate model parameters by measuring their in-distribution performance and model statistics for $B = 128$ samples (number of test-tasks). We report 2-sided confidence intervals with a confidence level $\alpha = 0.99$ for each metric X aggregated over the seeds r and the test-tasks B. For the full details on the experiment and baseline setup, see Appendix 2.B (code available here https://github.com/joeryjoery/lvrnn).

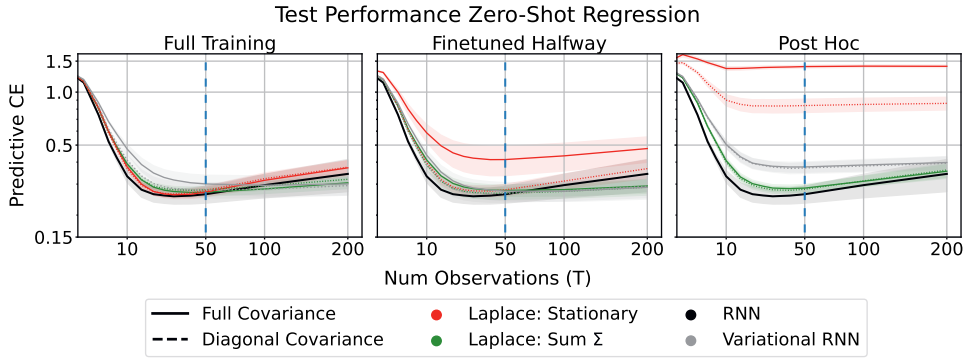


Figure 2.1: Final performance on the zero-shot regression task in terms of predictive cross-entropy, this should decrease over time. The left column shows results for complete model training, the middle and right columns perform model pre-training with the RNN (black). The middle column includes parameter finetuning, the right column does not. The blue dashed line indicates the training cut-off ($T = 50$).

2.5.1 Supervised Learning

As a didactic test-setup, we evaluated our method on noiseless 1D regression tasks. We generated data by sampling parameters to a Fourier expansion and then sampling datasets $\{\{(X_i, f_j(X_i))\}_{i=1}^T\}_{j=1}^n$ where each $X_i \sim \text{Unif}(-1, 1)$, $f_j \sim p_{\text{Fourier}}(f)$, and $n = 256, T = 50$. During training, we optimized a lower bound for a supervised domain using a weight for the KL-term of $\beta = 10^{-2}$ (see Eq. (2.2); Appendix 2.A.1). During testing, we computed the predictive cross-entropy (CE) with the true data-generating distribution and our model. So, at each step t, we used $H_t = \{X_i, f(X_i)\}_{i=1}^t$ to estimate the posterior predictive distribution $\mathbb{E}_{q_\theta(Z_t|H_t)} p_\theta(Y_i|X_i, Z_t)$ with Monte-Carlo using $m = 30$ samples. The predictive CE was estimated using Monte-Carlo over a large test dataset.

Variations Our Laplace VRNN used a stationary $q_\theta(Z_t|H_{<t})$ assumption (Laplace: Stationary; red) and a Markovian $q_\theta(Z_t|H_{t-1}, Z_{t-1})$ assumption (Laplace: Sum Σ; green) on the graphical model from Eq. (2.1). In practice, the stationary model computes the covariance for each datapoint in $H_{<t}$ at each t, whereas, the Markovian model sums the covariances for each pair (X_i, Y_i). To reduce clutter, we only show the Laplace VRNN ablation that sums the covariance, which also performed best among our variations (c.f., Appendix 2.C). We tested both diagonal and full covariance matrices.

Results As shown in Figure 2.1, across our comparisons the predictive CE goes down initially (except for the post-hoc stationary Laplace VRNN), however slightly increases again after the size of the dataset exceeds that seen during training $T > 50$. Notably, we see that our stationary Laplace VRNN strongly degrades performance when not used at the beginning (red), most notably the full-covariance variation. This could be an indication that the Laplace approximated posterior is too wide, whereas the point-estimate is extremely sharp, causing samples from our method to be out-of-distribution for the predictive model. In contrast, this result also shows that our method with the Markovian assumption (green)

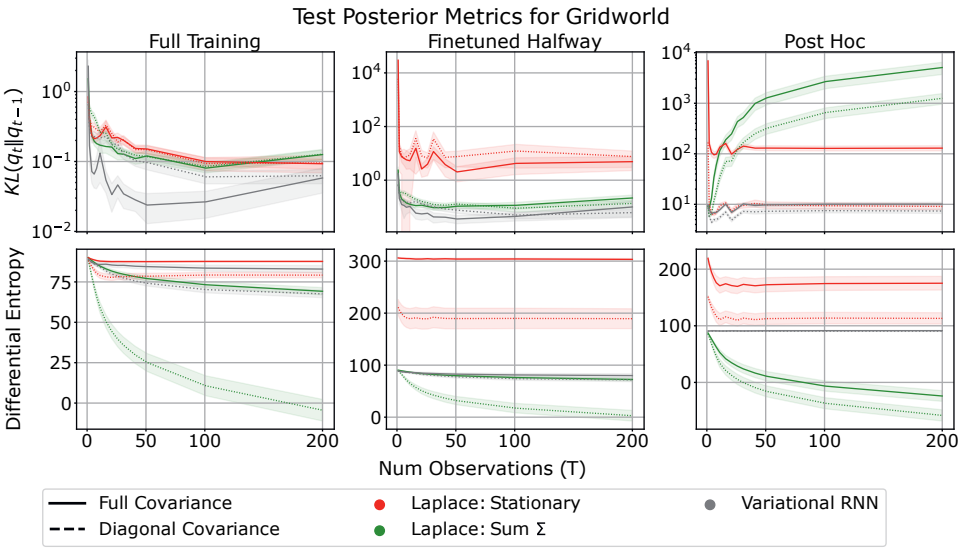


Figure 2.2: Evolution of summary statistics for the posterior model during testing. The top row shows the KL-divergences between consecutive posteriors q_t and q_{t+1}, and the bottom row shows the model entropy over time. In principle, we expect all lines to decrease gradually with more observations. When applying our Laplace approximation with summed covariances (green) after deterministic pre-training (right-column), we see that the posterior becomes more and more confident but does not converge to a stable distribution.

performs at least as well as the baselines in all cases while also providing a Bayesian posterior for the RNN after training.

2.5.2 Reinforcement Learning

To show that our method can perform uncertainty quantification while maintaining strong performance in reinforcement learning problems, we evaluated our method on a stochastic 5-armed bandit and a deterministic 5×5 gridworld with sparse rewards. We tested all models using a variant of recurrent PPO (Schulman et al., 2017) as a simple approximation for Eq. (2.4). During training, we used a batch size of $B = 256$ task samples and a sequence length of $T = 50$ interactions with the bandit and $T = 100$ for the gridworld task.

For the bandit, we generated training tasks by sampling reward probabilities from a Dirichlet prior using $\vec{\alpha} = 0.2$. In this domain we only condition our policy on the sampled model hypotheses $z_t \sim q_\theta(Z_t|H_t)$, $a_t \sim \pi_\theta(Z = z_t)$ as is typical in Thompson sampling (Osband et al., 2013). This experiment also aimed to investigate robustness to model sampling noise. For the gridworld (Zintgraf et al., 2021) we sampled tasks by generating the agent's start- and goal-tile uniformly randomly across the grid. In contrast to the bandit problem, the gridworld agent modeled the task as a Bayes-adaptive Markov decision process (Duff, 2002). Meaning that the policy conditions on both the model samples Z and the current state, $a_t \sim \pi_\theta(Z = z_t, S = s_t)$.

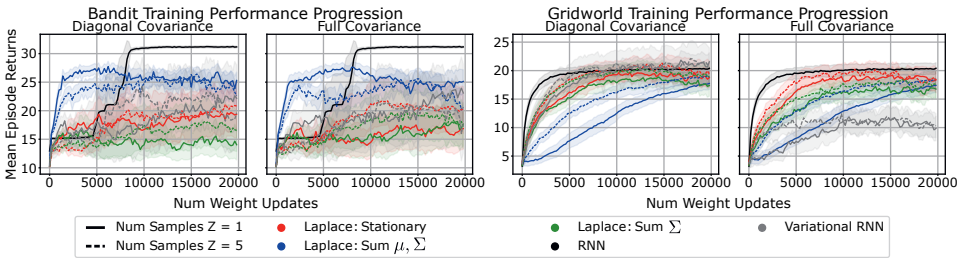


Figure 2.3: Average return curves during training for the Reinforcement Learning experiments. The dashed and solid lines (Num Z) indicate the number of Monte-Carlo samples used for the posterior model during training inside the modified lower bound of Eq. (2.4), to validate off-policy robustness in our loss. As expected, the deterministic RNN performs best, but our Laplace VRNN also outperforms the baseline VRNN.

Variations As before, we test different assumptions for our Laplace VRNN agent's model from Eq. (2.1). In this instance, we used a *windowed* version of the stationary Laplace VRNN $q_\theta(Z_t|H_{t-w-1:t-1})$ for $w = 10$ (red). I.e., this truncates the history up to a certain timestep to improve the runtime of the covariance computation, which otherwise scales in $\mathcal{O}(t)$-time. We also tested two variations of the Markovian $q_\theta(Z_t|H_{t-1}, Z_{t-1})$ factorization, which scaled in $\mathcal{O}(1)$-time. The proper-Markovian method (blue) sums the mean and covariance computed at each state-action (S_t, A_t) whereas the second variant only sums the covariance (green).

Following our reasoning from this chapter's introduction, we expect the deterministic RNN baseline to achieve the highest training returns, as its objective maximizes expected returns without the variational regularization on its latent state (Eq. (2.4)) and it incurs no sampling noise for Z_t. This also reserves model capacity for the policy and value networks, without the need to maintain a full (latent) posterior model (Goyal et al., 2017, Greenberg et al., 2023, Hafner et al., 2019). As a consequence, however, our hypothesis is that this variant produces less informative posterior statistics through our method, since its latent state is never explicitly trained as a posterior.

Results We visualize the evolution of estimated posterior statistics during testing in figure 2.2, where we removed the ablation that sums both the mean and covariance (blue) to reduce clutter (this ablation performed in between the other two, see Appendix 2.C). We plotted the differential entropy of the posteriors q_θ and the consecutive KL-divergences $KL(q_\theta(Z_t|...)\|q_\theta(Z_{t-1}|...))$ between posteriors over time, to see whether their behavior matches that of the true posterior. For the true posterior, we expect the entropy to decrease gradually with more observations T, which indicates that our model concentrates around some true value. Furthermore, the KL-divergences should converge to zero.

As expected, we see that the posterior entropy of the Bayesian methods reliably goes down and the KL-divergences gets close to zero (left-column). We see a similar pattern when doing finetuning (middle-columns) except for our stationary Laplace variation (red). Most importantly, we see a strong effect of our accumulating covariances variation (green) when using a post-hoc posterior approximation. We see that the entropy steadily decreases while the KL-divergences between consecutive posteriors grows larger and larger. In

contrast, the post-hoc VRNN (grey) and stationary Laplace (red) stay relatively constant, and are therefore non-informative. This result shows that the deterministic RNN does not converge to a stable hidden state when not explicitly regularized during training. This means that the learned estimator becomes more and more confident while not being consistent (Xiong et al., 2021).

The average training returns for our model ablations are shown in Figure 2.3. As argued in the introduction, we find that the deterministic method (black) has strong performance while also being the least noisy in the mean episode returns and being the fastest to train in terms of algorithm runtime. Interestingly, the proper-Markovian factorization (blue) of our Laplace VRNN showed faster learning in the Bandit up to a certain point, whereas it degraded training performance for the gridworld. All Bayesian methods tested on the bandit task were significantly noisy and only achieved sub-linear cumulative regret during test-time about 50% over all experiment repetitions. On the grid task, all methods achieved sub-linear cumulative regret.

In summary, none of the ablations for our Bayesian methods degraded performance when applied after deterministic pre-training without finetuning (post-hoc). Our Laplace VRNN also typically performed on par with the VRNN in terms of returns. However, only our Markovian Laplace variation that summed the covariances (green) could produce insightful posterior statistics of the pre-trained model. This confirms all our research questions of whether our proposed Laplace VRNN, and what model assumptions, can give useful posterior statistics while not degrading performance.

2.6 Conclusions

We have described how the Laplace approximation can be applied to recurrent neural network models in a zero-shot meta-reinforcement learning context. Our method is a cheap transformation of an existing recurrent network to a Bayesian model. This enables trained agents to more accurately model their task-uncertainty which can be used to create better or more robust methods.

We tested our method on supervised and reinforcement learning tasks to investigate the utility of our approximation (the quality of posterior statistics), how it depends on model assumptions (ablations), and how it compares against variational inference or point-estimate baselines (no degradation in performance). Our results show that the proposed *Laplace variational recurrent neural network* can reliably transform existing non-Bayesian models to produce a Bayesian posterior, at any point during training without modifying the model or training procedure. In contrast, variational inference requires altering the model architecture and training setup despite matching (or underperforming) compared to our method.

One limitation of our method is the computation of the Jacobians and possible restrictiveness of the Gaussian distribution. Furthermore, the RNN based agents required a variety of simplifications to our probabilistic model formulation (at least, for some of the tested configurations). Although the results matched expected behavior, and is consistent with prior work (Mikulik et al., 2020, Xiong et al., 2021), future work should investigate what the induced biases entail for our approximated posterior. Our method also does not

fix the overconfidence of the non-Bayesian agents, but enables us to observe this effect. Extending our approach to enable statistically sound representation learning or to improve exploration through e.g., distribution-shift detection in meta-RL are a promising directions for further study (Daxberger et al., 2021).

2

Appendix

2.A Derivations

2.A.1 Lower Bounds

In this section, we derive the two lower bounds used for model training in the main paper. These lower bounds are not particularly new or special, they only show how one can derive a learning objective from a probabilistic graphical modeling perspective.

Supervised Learning

For the supervised learning domain we can derive an evidence lower bound on the data-marginal as a training objective for our neural network parameters in the following way. For all permutations of $H^{1:n} = \{X^i, Y^i\}_{i=1}^n$, where $X^i \in \mathcal{X}, Y^i \in \mathcal{Y}$, we have,

$$p(H^{1:n}) = \int p(H^{1:n}|z)p(z)dz \tag{2.11}$$

$$= \int p(X^n, Y^n|z, H^{<n})p(H^{<n}|z)p(z)dz \tag{2.12}$$

$$= \int p(X^n, Y^n|z)p(H^{<n}|z)p(z)dz \tag{2.13}$$

$$= p(H^{<n}) \int p(X^n, Y^n|z)p(z|H^{<n})dz \tag{2.14}$$

if we complete the recursion for $p(H^{<n})$ and do importance sampling on the posterior with q, we get,

$$\ln p(H^{1:n}) = \ln \prod_{i=1}^{n} \int p(X^i, Y^i|z^i)p(z^i|H^{1:i})dz^{1:n} \tag{2.15}$$

$$= \sum_{i=1}^{n} \ln \int p(X^i, Y^i|z^i) \frac{q(z^i|H^{1:i})}{q(z^i|H^{1:i})} p(z^i|H^{1:i})dz^{1:n} \tag{2.16}$$

$$\geq \sum_{i=1}^{n} \int q(z^i|H^{1:i}) \left[\ln p(X^i, Y^i|z^i) + \ln \frac{p(z^i|H^{1:i})}{q(z^i|H^{1:i})} \right] dz^{1:n} \tag{2.17}$$

$$= \sum_{i=1}^{n} \mathbb{E}_{q(Z^i|H^{1:i})} \ln p(X^i, Y^i|Z^i) - KL(q(Z^i|H^{1:i}) \| p(Z^i|H^{1:i})), \tag{2.18}$$

which gives us the lower-bound for our approximate inference model when we use neural network parameters θ for the predictive and posterior models (overloading notation for q_ϕ

in Eq.(2.3)),

$$\mathcal{L}(\theta, H^{1:n}) = \sum_{i=1}^{n} \mathbb{E}_{q_\theta(Z^i|H^{1:i})} \underbrace{\ln p_\theta(X^i, Y^i|Z^i)}_{\text{Prediction Loss}} - \beta \cdot \underbrace{KL(q_\theta(Z^i|H^{1:i}) \| \texttt{stop_grad}[q_\theta(Z^{i-1}|H^{1:i-1})])}_{\text{Complexity Penalty}}, \quad (2.19)$$

where `stop_grad`$[\cdot]$ indicates a stop-gradient operation and $\mathcal{L}$ should be *maximized* with respect to θ. The hyperparameter $\beta \in \mathbb{R}_+$ accounts for differences in scaling. The stop-gradient is necessary so that the posterior at time t does not depend on the future. During generation and training, we also assume a uniform prior over the inputs $p(X^i|Z) = \text{Unif}$. Of course, this is just one lower bound, the one we use in the main paper for the reinforcement learning tasks also assumes that each z^i is sequentially dependent. In this case, the product would appear inside the integral in the first line for $\ln p(H^{1:n})$, this is only relevant for the Laplace VRNN that accumulates the mean and covariances.

For simplicity, we only perform training on a single permutation of H^n (i.e., canonical order), as in expectation all permutations are covered anyway and this provides training batches with more diverse examples. Unfortunately, when amortizing the computation of this lower bound with recurrent models it can be difficult to properly distill this permutation invariance of the data into the model. Using a recurrent model that linearly transforms the state, like a transformer (Katharopoulos et al., 2020, Vaswani et al., 2017) or general state space model (Bishop, 2007), would prevent this. We leave this open for future work.

Reinforcement Learning

Consider the joint distribution over environment traces $H^i = \{S_t, R_t, A_t\}_{t=1}^T$ and latent variables Z, we'll write episode indices (*extra*-episodic) $i = 1 \ldots, n$, in the superscript and time indices (*inter*-episodic) $t = 1, \ldots, T$, in the subscript,

$$p(H^{1:n}, Z^{1:n}) = \prod_{i=1}^{n} p(H^i, Z^i|H^{<i}, Z^{<i}) \quad (2.20)$$

$$= \prod_{i=1}^{n} \prod_{t=1}^{T_i} p(S_t^i, R_t^i, A_t^i, Z_t^i|S_{<t}^i, R_{<t}^i, A_{<t}^i, Z_{<t}^i, Z^{<i}, H^{<i}) \quad (2.21)$$

$$= \prod_{i=1}^{n} \prod_{t=1}^{T_i} p(S_t^i, R_t^i, A_t^i|Z_t^i, H_{<t}^i) p(Z_t^i| \underbrace{Z_{<t}^i, H_{<t}^i}_{inter}, \underbrace{Z^{<i}, H^{<i}}_{extra}) \quad (2.22)$$

$$= \prod_{i=1}^{n} \prod_{t=1}^{T_i} \underbrace{p(R_t^i|S_t^i, A_t^i, Z_t^i)}_{\text{Reward Model}} \underbrace{\pi(A_t^i|S_t^i, Z_t^i)}_{\text{Action Model}} \underbrace{p(S_t^i|S_{t-1}^i, A_{t-1}^i, Z_t^i)}_{\text{Transition Model}} \underbrace{p(Z_t^i|Z_{<t}^i, H_{<t}^i, Z^{<i}, H^{<i})}_{\text{Posterior Model}}, \quad (2.23)$$

the lower-bound in Eq. 2.2 can then be easily derived by doing importance sampling on the posterior model with q, marginalizing out the latent variables, and assuming that H^i is

independent of all other variables given the latent-variable Z^i,

$$\ln p(H^{1:n}) = \ln \int \prod_{i=1}^{n} p(H^i, z^i | H^{<i}, z^{<i}) dz^{1:n} \tag{2.24}$$

$$= \ln \int \prod_{i=1}^{n} p(H^i|z^i) \frac{q(z^{1:i}|H^{<i})}{q(z^{1:i}|H^{<i})} p(z^i|H^{<i}, z^{<i}) dz^{1:n} \tag{2.25}$$

$$\geq \int \sum_{i=1}^{n} q(z^{1:i}|H^{1:i}) \left(\ln p(H^i|z^i) - \ln \frac{q(z^{1:i}|H^{<i})}{p(z^i|H^{<i}, z^{<i})} \right) dz^{1:n} \tag{2.26}$$

$$\propto \mathbb{E}_{q(Z^{1:n}|H^{1:n})} \sum_{i=1}^{n} \ln p(H^i|Z^i) - KL(q(Z^i|Z^{<i}, H^{<i}) \| p(Z^i|Z^{<i}, H^{<i})). \tag{2.27}$$

As stated in the paper, this lower bound only reproduces the data but does not maximize the rewards per se. So, using the control as inference framework (Levine, 2018), if we write the conditional that a given trajectory H is desirable as $p(\mathcal{O} = 1|H) \propto \exp(\sum_{t=1}^{T} R_t)$, then we can derive a lower bound for the sampling distribution for a reinforcement learning agent as (again simplifying notation for $\mathcal{L}$ compared to the main text),

$$\ln p(\mathcal{O} = 1) = \ln \mathbb{E}_{q(H^{1:n}, Z^{1:n})} p(\mathcal{O} = 1|H^{1:n}, Z^{1:n}) \frac{p(H^{1:n}, Z^{1:n})}{q(H^{1:n}, Z^{1:n})} \tag{2.28}$$

$$\geq \mathbb{E}_{q(H^{1:n}, Z^{1:n})} \ln p(\mathcal{O} = 1|H^{1:n}) - KL(q(H^{1:n}, Z^{1:n}) \| p(H^{1:n}, Z^{1:n})), \tag{2.29}$$

$$= \mathcal{L}(q) \tag{2.30}$$

where we define the variational distribution $q(H^{1:n}, Z^{1:n})$ to factorize in exactly the same way as $p(H^{1:n}, Z^{1:n})$ where we fix the reward and transition models and then modify the action and posterior models. This choice of factorization cancels out the fixed terms in the KL-divergence, giving us the lower bound (Eq.(2.4) in the main-text),

$$\mathcal{L}(q) = \mathbb{E}_{q(H^{1:n}, Z^{1:n})} \sum_{i=1}^{n} \sum_{t=1}^{T_i} R_t^i - KL\left(q(A_t^i|S_t^i, Z_t^i) \| \pi(A_t^i|S_t^i, Z_t^i)\right) \tag{2.31}$$

$$- KL\left(q(Z_t^i|Z_{<t}^i, H_{<t}^i, Z^{<i}, H^{<i}) \| p(Z_t^i|Z_{<t}^i, H_{<t}^i, Z^{<i}, H^{<i})\right). \tag{2.32}$$

To amortize computation of this lower bound and make this practical to compute, we parametrize the variational posterior $q_\theta(Z|\dots)$ and action model $\pi_\psi(A|\dots)$. To then finally give us a practical optimization objective for the parameters θ, ψ, we substitute for the action model $\pi(A|\dots) = \pi_{\psi_{old}}$ and for the true posterior we simply use $q_\theta(Z|\dots)$ with a stop-gradient $\square$. We scale the KL-penalty with a hyperparameter $\beta \in \mathbb{R}_+$ to account for differences in scaling. This gives us our final lower-bound,

$$\mathcal{L}(\theta, \psi) = \mathbb{E}_{q_\theta(H^{1:n}, Z^{1:n})} \sum_{i=1}^{n} \sum_{t=1}^{T_i} R_t^i - KL\left(\pi_\psi(A_t^i|S_t^i, Z_t^i) \| \pi_{\psi_{old}}(A_t^i|S_t^i, Z_t^i)\right) \tag{2.33}$$

$$- \beta \cdot KL\left(q_\theta(Z_t^i|Z_{<t}^i, H_{<t}^i, Z^{<i}, H^{<i}) \| \square q_\theta(Z_{t-1}^i|Z_{<t-1}^i, H_{<t-1}^i, Z^{<i}, H^{<i})\right),$$

which we can plug into the l.h.s. of the amortization objective in Eq. (2.3) enabling us to optimize our model parameters through sampling and end-to-end differentiation from the

policy to the posterior. Although this is the objective we desire, we make further heuristic approximations through the use of the Proximal Policy Optimization algorithm (Schulman et al., 2017). This could roughly be interpreted as doing expectation-propagation (Bishop, 2007) on the policy (i.e., swapping the KL-arguments for the policies).

Our lower bound is an extension of the one by Abdolmaleki et al. (2018), for standard Markov decision processes, to include the latent variable posterior for use in memory-based meta-reinforcement learning (Duan et al., 2016). When using an RNN to approximate the posterior, the KL-penalty for $q_\theta(Z|\dots)$ is typically ignored since this is undefined for point-estimates. Doing this would recover the RL^2 objective in combination with MPO (Abdolmaleki et al., 2018, Duan et al., 2016).

Posterior Factorization

To choose an efficient factorization for our variational model we need the following result,

Lemma 2.1. *We can write* $p(Z|\{X_i\}_{i=1}^n) = \frac{1}{p(Z)^{n-1}} \prod_{i=1}^n p(Z|X_i)$ *iff* $X_i \perp\!\!\!\perp X_j, \forall j \neq i$.

Proof. This result can be shown by applying Bayes rule then factorizing each X_i to be independent of $X_j, \forall j \neq i$ and then applying Bayes rule again,

$$
\begin{aligned}
p(Z|\{X_i\}_{i=1}^n) &= \frac{p(X_1, X_2, \dots, X_n|Z)p(Z)}{p(X_1, X_2, \dots, X_n)} && (2.34)\\
&= \frac{p(Z)}{\prod_{i=1}^n p(X_i)} \prod_{i=1}^n p(X_i|Z) && \text{(Independence)}\\
&= \frac{p(Z)}{\prod_{i=1}^n p(X_i)} \left[\prod_{i=1}^n p(Z|X_i) \frac{p(X_i)}{p(Z)} \right] && (2.35)\\
&= \frac{p(Z)}{\prod_{i=1}^n p(X_i)} \left[\frac{\prod_{i=1}^n p(X_i)}{p(Z)^n} \prod_{i=1}^n p(Z|X_i) \right] && (2.36)\\
&= \frac{1}{p(Z)^{n-1}} \prod_{i=1}^n p(Z|X_i) && (2.37)
\end{aligned}
$$

□

2.A.2 Laplace Variational Recurrent Model

Proposition 2.2. *Given a mean-field assumption on the data for our posterior* q_θ *(Lemma 2.1). The second order Taylor Expansion of* $\ln q_\theta(Z_t|H_{<t}, \phi_t)$ *linearized at* ϕ_t*, where* $\phi_t = \phi^*$ *is a local maximizer of* q_θ *and occupies the same space as* Z_t*, yields the following Gaussian distribution,*

$$
q_\theta(Z_t|H_{<t}, \phi_t) = \mathcal{N}\left(Z_t; \mu = \phi_t, \Sigma = (-\nabla^2_\phi \ln q_\theta(Z_t|H_{<t}, \phi)|_{\phi=\phi_t})^{-1}\right). \tag{2.38}
$$

Proof. Reiterating the results from the main paper, we choose to factorize our model as,

$$q_\theta(Z_t|H_{<t},\phi_t) = \frac{1}{q_\theta(Z_t|\phi_t)^{t-2}}\prod_{i=1}^{t-1} q_\theta(Z_t|S_i,R_i,A_i,\phi_t), \tag{Lemma 2.1}$$

$$= \exp\left[(2-t)\ln q_\theta(Z_t|\phi_t) + \sum_{i=1}^{t-1}\ln q_\theta(Z_t|S_i,R_i,A_i,\phi_t)\right] \tag{2.39}$$

$$= \exp h_\theta(Z_t;H_{<t},\phi_t), \tag{2.40}$$

where our aim is to make a local approximation to h_θ, the rest of the proof follows Appendix A from Daxberger et al. (2021).

The second order Taylor expansion of $h_\theta(Z_t;H_{<t},\phi_t)$ where ϕ_t (locally) maximizes h_θ keeping all other arguments fixed, gives us,

$$\hat{h}_\theta(Z_t;H_{<t},\phi) = h_\theta(Z_t;H_{<t},\phi_t) + \underbrace{\nabla_\phi h_\theta|_{\phi=\phi_t}(\phi-\phi_t)}_{=0} + \frac{1}{2}(\phi-\phi_t)^\top\nabla^2_\phi h_\theta|_{\phi=\phi_t}(\phi-\phi_t),$$

$$= h_\theta(Z_t;H_{<t},\phi_t) - \frac{1}{2}(\phi-\phi_t)^\top\nabla^2_\phi h_\theta|_{\phi=\phi_t}(\phi-\phi_t), \tag{2.41}$$

dropping the function arguments to h_θ for the higher-order terms for brevity. When exponentiating $\hat{h}_\theta$ and renormalizing it to integrate to 1, it is easy to show that we recover a Gaussian,

$$h_\theta(Z_t;H_{<t},\phi_t) \approx \frac{1}{\int_{\mathbb{R}}\exp\hat{h}_\theta(Z_t;H_{<t},\phi')d\phi'}\exp\hat{h}_\theta(Z_t;H_{<t},\phi) \tag{2.42}$$

$$= \frac{\exp\{h_\theta(Z_t;H_{<t},\phi_t) - \frac{1}{2}(\phi-\phi_t)^\top(-\nabla^2_\phi h_\theta|_{\phi=\phi_t})(\phi-\phi_t)\}}{\int_{\mathbb{R}}\exp\{h_\theta(Z_t;H_{<t},\phi_t) - \frac{1}{2}(\phi'-\phi_t)^\top(-\nabla^2_\phi h_\theta|_{\phi=\phi_t})(\phi'-\phi_t)\}d\phi'} \tag{2.43}$$

$$= \frac{\exp\{-\frac{1}{2}(\phi-\phi_t)^\top(-\nabla^2_\phi h_\theta|_{\phi=\phi_t})(\phi-\phi_t)\}}{\int_{\mathbb{R}}\exp\{-\frac{1}{2}(\phi'-\phi_t)^\top(-\nabla^2_\phi h_\theta|_{\phi=\phi_t})(\phi'-\phi_t)\}d\phi'} \tag{2.44}$$

$$= \mathcal{N}\left(Z;\mu=\phi_t,\Sigma=(-\nabla^2_\phi\ln q_\theta(Z_t|H_{<t},\phi)|_{\phi=\phi_t})^{-1}\right). \tag{2.45}$$

□

Observe that the above result technically gives us a distribution over ϕ, and misleadingly not Z. However, this is simply a consequence of our notation and distributional assumptions for q. In practice, ϕ is computed by our representation model (recurrent neural network), and our probabilistic model conflates the learned representation ϕ as the mode of the distribution for the latent distribution Z.

Proposition 2.3. *If we choose $q_\theta(Z_t|S_i,R_i,A_i,\phi) = \mathcal{N}(Z_t;\mu=f_\theta(S_i,R_i,A_i;\phi),\Sigma=I_n)$ and $q_\theta(Z_t|\phi) = \mathcal{N}(Z_t;\mu=\phi,\Sigma=\sigma^2_\phi I_n)$ where we take the limit for σ^2_ϕ to infinity, then our Laplace approximated posterior (Proposition 2.2) has an inverse covariance that is computed as,*

$$\Sigma_t^{-1} = \sum_{i=1}^{t-1}(\nabla_\phi f_\theta(S_i,R_i,A_i;\phi)|_{\phi=\phi_t})(\nabla_\phi f_\theta(S_i,R_i,A_i;\phi)|_{\phi=\phi_t})^\top. \tag{2.46}$$

Proof. To see this we only need to write down the Hessian under a local maximum assumption of $\phi = \phi^*$ (Laplace approximation) and substitute the chosen Gaussian distributions in for all terms.

$$\nabla^2_\phi \ln q_\theta(Z_t|H_{<t},\phi) = \nabla^2_\phi \left[(2-t)\ln q_\theta(Z_t|\phi) + \sum_{i=1}^{t-1} \ln q_\theta(Z_t|S_i,R_i,A_i,\phi)\right] \tag{2.47}$$

$$= \nabla^2_\phi \left[(2-t)\ln \mathcal{N}(Z_t;\phi,\sigma^2_\phi I_n) + \sum_{i=1}^{t-1} \ln \mathcal{N}(Z_t; f_\theta(S_i,R_i,A_i;\phi), I_n)\right] \tag{2.48}$$

$$= \frac{t-2}{\sigma^2_\phi} I_n + \sum_{i=1}^{t-1} (J_\phi)_i \underbrace{\left(\nabla^2_\mu \ln \mathcal{N}(Z_t;(f_\theta)_i, I_n)\right)}_{=-1} (J_\phi)_i^\top + \underbrace{\sum_{i=1}^{t-1} \nabla^2_\phi (f_\theta)_i|_{\phi=\phi_t} \nabla_\mu \ln \mathcal{N}(Z_t;(f_\theta)_i, I_n)}_{=0, \text{ when } \phi_t=\phi^* \text{ (Laplace approx.)}} \tag{2.49}$$

$$= \frac{t-2}{\sigma^2_\phi} I_n - \sum_{i=1}^{t-1} (J_\phi)_i (J_\phi)_i^\top, \tag{2.50}$$

where we abbreviate $(J_\phi)_i = \nabla_\phi f_\theta(S_i,R_i,A_i;\phi)$ and $(f_\theta)_i = f_\theta(S_i,R_i,A_i;\phi)$. Then, in the case of using an infinite variance Gaussian for the prior, $\lim_{\sigma^2_\phi \to \infty} \nabla^2_\phi \ln q_\theta(Z_t|H_{<t},\phi)$, we get,

$$\Sigma_t^{-1} = \left(-\nabla^2_\phi \ln q_\theta(Z_t|H_{<t},\phi)|_{\phi=\phi_t}\right) = \sum_{i=1}^{t-1} (J_\phi)_i (J_\phi)_i^\top \tag{2.51}$$

$$= \sum_{i=1}^{t-1} (\nabla_\phi f_\theta(S_i,R_i,A_i;\phi)|_{\phi=\phi_t})(\nabla_\phi f_\theta(S_i,R_i,A_i;\phi)|_{\phi=\phi_t})^\top. \tag{2.52}$$

□

Final Model

To complete our fully general Laplace approximated variational recurrent neural network, we need to define the posterior over *all* previous latent variables, $q_\theta(Z_t|Z_{<t},H_{<t})$. We can easily plug this dependency in for our Laplace approximation from Prop. 2.2 by assuming that each consecutive posterior has an additive effect on all future posteriors (i.e., a Gaussian convolution),

$$q_\theta(Z_t|Z_{<t},H_{<t}) = \mathcal{N}\left(Z_t; \mu_t = \phi_t + \sum_{i=1}^{t-1} Z_i, \Lambda_t = -\nabla^2_\phi \ln q_\theta(Z_t|Z_{<t},H_{<t},\phi_t)|_{\phi=\phi_t}\right). \tag{2.53}$$

As long as we do not condition ϕ_t on $Z_{<t}$, the dependency on past latent variables becomes a constant w.r.t. ∇^2_ϕ, making the covariance independent of these terms. This is in contrast to a recurrent state-space model architecture which does condition on these values (Hafner et al., 2020), however, our choice permits an analytical solution. It is a known result that the

expected posterior then becomes another Gaussian with the means and inverse covariances summed (Bromiley, 2003),

$$\begin{aligned}&\mathbb{E}_{q_\theta(Z_{<t}|H_{<t})} q_\theta(Z_t|Z_{<t}, H_{<t}) \\ &= \mathcal{N}\left(\mu_t = \phi_t + \sum_{i=1}^{t-1} \mu_i, \Lambda_t = -\nabla^2_\phi \ln q_\theta(Z|H_{<t}, \phi)|_{\phi=\phi_t} + \sum_{i=1}^{t-1} \Lambda_i\right). \end{aligned} \tag{2.54}$$

This particular form has also been described by Ritter et al. (2018) in the context of continual learning. It is easy to accumulate the mean and covariances terms sequentially over t. Depending on the assumptions one makes on the data-generating distribution, one can sum over fewer terms to make the calculation more efficient. The ones we ran experiments for in the main paper include:

1. **Stationary Posterior**: $q_\theta(Z_t|H_{<t})$. Full summation over $H_{<t}$ in the calculation of Λ_t. No summation over previous posteriors $Z_{<t}$.

2. **Markov Chain Posterior**: $q_\theta(Z_t|H_{t-1}, Z_{t-1})$. The inverse covariance is calculated only using the most recent observation $t-1$. Only the previous posterior mean and precision are summed with the current mean and precision.

3. **Windowed Markov Chain Posterior**: $q_\theta(Z_t|H_{k:t-1}, Z_{t-1})$. The inverse covariance is calculated using the k most recent observations. Only the previous posterior mean and precision are summed with the current mean and precision.

However, these are all simplified models, whereas the variational recurrent model (Chung et al., 2015) we discuss in the main paper is fully general.

On summation of the means The current formulation for the probabilistic model mismatches with typical recurrent neural network (RNN) architectures. Typically, RNN models do not aggregate their past hidden states for the outputs. However, as discussed in the main paper, we strictly required this simplification for the Markov chain (non-stationary) factorizations, since this was the only approach that would leave the base RNN architecture untouched. Despite that, we can make two heuristic arguments that can partially explain the effect of summing all the previous covariances but omit summation of the mean:

- **Representation Learning:** When using covariance summation throughout training (no post-hoc posterior, or finetuning of deterministic baselines), an RNN can learn to represent the hidden state aggregation implicitly within the state-update.

- **Exponential Tilting:** Omitting the mean summation can be interpreted as a form of exponential tilting of Gaussian distributions. This is an importance-sampling technique for rare-event simulation (Asmussen & Glynn, 2007). For the correctly chosen tilting parameter, this can have the same effect as subtracting all previous means (at the cost of some bias).

2.B Implementation Details

2.B.1 Model Architecture and Optimization

Following the main text we can define our model according to the following components,

$$
\begin{aligned}
&\text{Embedding} && S_t^g, A_t^g, R_t^g = g_\theta(S_t), h_\theta(A_t), w_\theta(R_t), \\
&\text{Recurrent Model} && \phi_{t+1} \quad = f_\theta(S_t^g, A_t^g, R_t^g; \phi_t), \\
&\text{Posterior Model} && Z_t \quad \sim q_\theta(Z_t | H_{<t}, \phi_t), \\
&\text{Reward Model} && \hat{R} \quad \sim p_\theta(\hat{R} | Z_t, A^g, S_t^g), \\
&\text{Action Model} && A_t \quad \sim \pi_\theta(A_t | Z_t, S_t^g),
\end{aligned}
$$

note that we do not train an action model for the supervised experiments, and we do not use the reward model in the reinforcement learning experiments (i.e., we do not use it to select actions or to do planning). In the supervised case, the reward model simply learns a direct function prediction $\hat{R}$ where the state can be considered stationary. For brevity, we denoted the full set of parameters as θ, in practice each component has its parameters but we jointly optimize for these using end-to-end differentiation.

For the embedding model we used a multi-layered perceptron (MLP) (Cybenko, 1989) of two hidden layers of width 256 nodes, we used leaky-ReLU for the activation. The action and predictive model used a three hidden layer MLP with sizes (256, 256, 64), also with leaky-ReLU. For the recurrent model, we use a long-short-term memory module (Hochreiter & Schmidhuber, 1997) with $n = 128$ hidden nodes. We projected the outputs (not the carried state) of the LSTM to a smaller $n = 64$ feature output vector with a learned affine transform (i.e., a 1-layer MLP without an activation).

For discrete environments, the action model predicted the n logits for the full action space. For the regression task, the reward model outputs a Gaussian with a learned mean and input-independent variance.

As noted in the main paper, we apply stop-gradients on the prior term appearing in the KL-divergences (Eq. 2.2 and Eq. 2.4). Doing this prevents past posteriors from fitting to data beyond their respective timestep, while still constraining our current posterior to not deviate too much from these terms. We also apply stop-gradients to the *past* mean and covariance terms when *accumulating* these terms. This is only relevant for the Laplace VRNN ablations. The reasoning for this is the same as for the stop-gradient in the KL term, we want to use the past means and covariances as constants at timestep t, and not as another learnable parameter. As a side note, we also found that doing this sped up training orders of magnitude.

We developed everything discussed in this paper in Jax v.0.4.23 (Bradbury et al., 2018). For neural network design, we used the Flax library v0.8.1 (Heek et al., 2024). For optimization, we used the Adamw optimizer (Loshchilov & Hutter, 2019) implemented in Optax with learning-rate = 10^{-3}, weight-decay = 10^{-6}, and the rest on default settings at version v0.1.7. We used gradient-clipping to have a max global norm of 1.0 and a maximum individual gradient of [-5.0, 5.0]. We used our implementation for the PPO algorithm with help from the RLax library to compute the generalized advantage estimators. For a full list of dependencies, requirements, and program flags, our code will be made available upon

publication.

2.B.2 Variational Posterior Baseline

As discussed in the main paper, typically in meta-reinforcement learning we see that the posterior $q_\theta(Z_t|H_{<t})$ is modeled with a point-estimate and we propose to use the Laplace approximation to convert this point-estimate into a Gaussian distribution. Instead of computing the covariance matrix using the Laplace approximation, we can also directly predict this covariance with another neural network.

So, our variational recurrent neural network (VRNN) baseline predicted the $m(m-1)$ lower triangular elements of the $m \times m$ dimensional covariance matrix (or just m for the diagonal ablations) and the m dimensional mean. We opted for a spectral decomposition $\Sigma = USV$ where S is diagonal and U was predicted through the Cayley map starting from a skew-symmetric matrix. First, we compute ϕ_t with our RNN, then we project ϕ_t to μ_t, S_t, L_t with a linear layer such that $\mu_t \in \mathbb{R}^m$, $S_t \in \mathbb{R}^{m \times m}$ is diagonal and $L_t \in \mathbb{R}^{m \times m}$ is lower-triangular, we then compute $U_t = (I - A_t)(I + A_t)^{-1}$ where $A_t = L_t - L_t^\top$ to get an orthogonal eigenbasis. We then construct the Gaussian as,

$$q_\theta(Z_t|H_{<t}, \phi_t) = \mathcal{N}(Z_t; \mu = \mu_t, \Sigma = U_t(\exp S_t)U_t^\top), \tag{2.55}$$

this representation also made matrix inversion incredibly easy as $(Ue^S V)^{-1} = Ue^{-S}V$ since $U = V^\top$ are orthogonal.

The reason for using the spectral decomposition was that we did not achieve stable training using any variant of the Cholesky factorization for the covariance matrix (LU or LDU decomposition), even if explicitly transforming the eigenvalues to be positive. We suspect that the spectral parameterization trained more stably than the LU or LDL parameterization as the basis matrices U or V are constrained to the group of orthogonal matrices. Therefore, each element in the predicted parameters L_n is also constrained. In contrast, the LDL parameterization leaves the triangular parameters in an unconstrained representation which might enable an unstable representation. However, we did not investigate this problem beyond what was needed to get our method working.

We also did not accumulate the mean or covariances for the VRNN, unlike for the Laplace VRNN, as this model parameterization could simply learn this function instead (or learn to undo this parameterization). The point of our experiments was not to squeeze performance out of our baseline but to have a strong reference for comparison. We also found that the VRNN was highly competitive with the Laplace VRNN.

2.B.3 Predictive Ensemble Averaging

Since we sample latent variables Z from our posterior q_θ, when we pass multiple samples through our predictive model or the action model, we obtain multiple distributions for the output modalities. Typically this induces a mixture distribution, however, we simply averaged out either the logits or the means and variances (C. Wang et al., 2020). This can be interpreted as a normalized Gaussian convolution over the output parameters or simply as a kind of bagging strategy over ensemble members $Z = z^{(i)}, i = 1, \ldots, k$.

The reasoning for opting for ensemble averaging instead of mixture distributions is

that this simply achieved more stable training in combination with PPO (Schulman et al., 2017). The main problem was caused by the penalty term of the policy entropy, our implementation did not achieve stable training when approximating this term in any way (so neither with mixtures nor with singular distributions), we only achieved stable training when the entropy term could be computed *exactly*.

We did not investigate in depth why an approximate entropy loss caused training divergence, but we speculate this is due to the problems of training on generated data as discussed in the context of language models by Shumailov et al. (2023). In this case, the tails of the action distribution slowly shrink over time as Monte-Carlo estimation of the entropy might not cover low-likelihood events sufficiently with small sample sizes. This phenomenon is referred to as *model collapse*, not to be confused with *posterior collapse* (Goyal et al., 2017). We suspect that this problem could be reduced by using a proper (approximate) Bayesian inference algorithm, like maximum a posteriori optimization (Abdolmaleki et al., 2018), which essentially removes the instability of reinforcement learning losses by casting policy learning as a supervised learning problem.

2.B.4 Environment Design

For our testing environments we implemented three problem domains, a 1D function regression problem (Finn et al., 2017), a discrete n-armed bandit problem (Duan et al., 2016), and an $n \times n$ open grid problem (Zintgraf et al., 2021), as shown in Figure 2.4. We generated many variations of these environments to learn from by sampling their dependent task parameters.

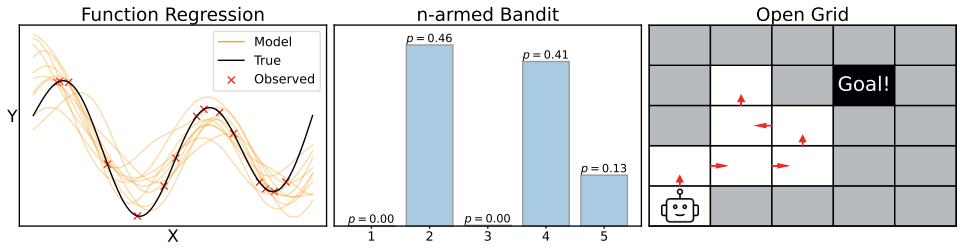


Figure 2.4: Visualization of sampled tasks we evaluated our method on. 1) Zero-shot learning of a function (left), 2) learning a stochastic best-arm selection algorithm (middle), and 3) learning a deterministic grid exploration agent (right).

Supervised. For the supervised problem we generated noiseless 1D test-functions on the bounded domain $[-1, 1]$. We did this through a Fourier expansion of $n = 4$ components where we randomly generate amplitudes, phase shifts, and input shifts. We manually tuned the ranges for these parameters and sampled uniformly random within these ranges. In other words, we can define the joint distribution over a function dataset as,

$$p(X, Y) = \delta(Y = \text{FourierSum}(Y; X, \varphi_{1:n}, c_{\text{shift}}, A_{0:n})) \cdot \text{Unif}(X; -1.0, 1.0), \tag{2.56}$$

$$c_{\text{shift}} \sim \text{Unif}(0.0, \pi), \varphi_i \sim \text{Unif}(0.0, \pi), A_i \sim \text{Unif}(-1.0, 1.0) \tag{2.57}$$

where the 'FourierSum' is computed in its amplitude-phase form. The training was performed with 50 examples per sampled function.

Bandit To generate bandit problems we used a Dirichlet distribution with $\alpha = 0.2$ during training and $\alpha = 0.3$ during testing (higher α makes the problem more difficult), this gave us a normalized vector of probabilities, $p_n \sim \text{Dir}(\alpha = 1_n \cdot 0.2)$ for which the agent needed to find $\max_i p_i$. Observations (which are equivalent to the rewards) were generated by sampling Bernoulli outcomes given p_i. Since each interaction of the agent with the bandit is seen as an episode, the environment returned discount factors of $\gamma = 0$. The training was performed over 50 total interactions.

Gridworld For the discrete grid environment we closely match the implementation of (Zintgraf et al., 2021). We reimplemented this environment in Jax to benefit from GPU acceleration for the data-generation process. The environment constructs a $n \times n$ open grid for the agent and uniformly randomly initializes the start and goal state (such that they don't overlap). The agent can choose between moving up, down, left, or right, the environment transitions deterministically but does not move the agent if it moves outside the bounds. The agent is rewarded with +1 if it encounters the goal and 0 otherwise, the observations were constructed as two one-hot-encoded vectors for the row and column index. The goal tile is *not* observed. If the agent did not find the goal within $T = 15$ steps it would be reset to its starting state and the discount factor would be set to $\gamma = 0$ at that transition. The training was performed over 100 total interactions.

2.B.5 Experimental Design and Hyperparameters

The supervised experiments did not provide additional hyperparameters to set, all necessary parameters were learned using an empirical cross-entropy with the data (see the main paper). For the reinforcement learning experiments we needed to set several hyperparameters for PPO (Schulman et al., 2017), these are given in Table 2.1. We did not use mini batching for our version of recurrent PPO and accumulated the loss over the full trajectories and batches. We found that larger batch sizes gave us faster and more stable learning.

Then our experimental design implied running an exhaustive parameter grid over the domain presented in Table 2.2. This grid was adjusted over successive experiments to reduce the computational footprint, this was manually tuned to select the parameter values that performed the best for both the baseline and our method. The parameter grid was applied equivalently on all problem domains for $r = 30$ distinct seeds. Although this experiment design is still quite modest, running this full grid induces 144 distinct configurations times 30 repetitions for the Laplace VRNN alone. One single run took on average $1\frac{1}{2}$ hour to complete for the gridworld environment on an A100 80GB NVIDIA GPU. Although, a better learning algorithm other than PPO (e.g., MPO), and minibatch optimizations could probably get the wallclock time down drastically while still achieving similar performance.

For the finetuning experiments we essentially made model snapshots of the deterministic baselines (RNNs) and reran the ablations as shown in Table 2.2 using the snapshots as starting weights. For each experiment, these snapshots were taken halfway, and three-quarters way during training in terms of the number of weight-updates.

Table 2.1: Proximal Policy Optimization loss parameters. Note that we use our Recurrent implementation for this algorithm.

Name	Symbol	Value
Minibatches		Full-Batch
Batch-Size		256
TD-Lambda	λ	0.9
Discount	γ	0.9
Policy-Ratio clipping	ϵ	0.2
Standardize Advantages		False
Exact Policy Entropy		True
Value Loss Scale		1.0
Policy Loss Scale		1.0
Entropy Loss Scale		0.1

To make some final informal notes on the choices for the parameters,

- We found that scaling the KL-penalties in the lower bound with hyperparameters that were slightly larger than $\beta > 10^{-2}$ caused posterior collapse (Goyal et al., 2017). Meaning, our posterior simply fitted its parameters to always match the prior despite accumulating more data.
- For the regression problem, using multiple samples for $z^{(i)}$ inside the lower bound decreased the predictive loss a lot. Showing that integration over the predictive is effective. Although, this did not result in better test-time performance.
- Using a small buffer size for the history window in the posterior $q_\theta(Z_t|H_{t-k:t-1})$ is more practical since the computation of the Jacobians is the main bottleneck of our method. We found that accumulation of only the covariance where each covariance is computed with a window of just 1, H_{t-1} results in the fastest method (in wallclock time) while being on par with many of the ablations.

Table 2.2: Proximal Policy Optimization loss parameters. Note that we use our recurrent implementation for this algorithm. All configurations were repeated for $r = 30$ repetitions (distinct random seeds). We also drop configurations that do not induce a valid model, e.g., $k_Z = 0$ and accumulation of Σ is not a valid configuration since there is no window to accumulate over.

Name	Symbol	Value
Deterministic RNN		
Posterior Dimensionality	n	$\{32, 64\}$
Variational RNN		
Posterior Dimensionality	n	$\{32, 64\}$
Covariance Parameterization		{Full, Diagonal}
Posterior KL-Penalty	β	$\{1.0, 10^{-2}, 10^{-4}\}$
Number of Posterior Samples	n_Z	$\{1, 5\}$
Laplace VRNN		
Posterior Dimensionality	n	$\{32, 64\}$
Covariance Parameterization		{Full, Diagonal}
Posterior KL-Penalty	β	$\{1.0, 10^{-2}, 10^{-4}\}$
Number of Posterior Samples	n_Z	$\{1, 5\}$
History Buffer Window	k_H	$\{1, 10\}$
Latent Variable Window	k_Z	$\{0, 1\}$
Accumulation		$\{(\mu, \Sigma), \Sigma\}$

2.C Supplementary Results



Figure 2.5: Progression of the predictive error during supervised model training of all ablations. Ablations are averaged over parameter groups as indicated by the legend. This figure does not show the finetuning results. To reduce computation time for subsequent results, we picked $\beta = 0.01$ for the reinforcement learning task ablations.

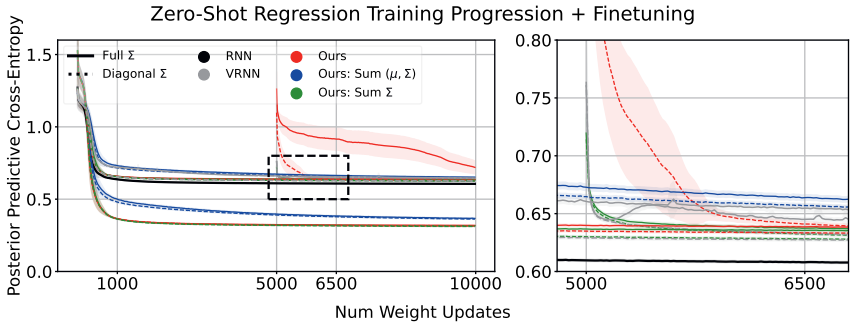


Figure 2.6: Zoom-in of Figure 2.5 for three finetune runs (left), for most ablations the predictive error quickly goes down to their full variational training error.

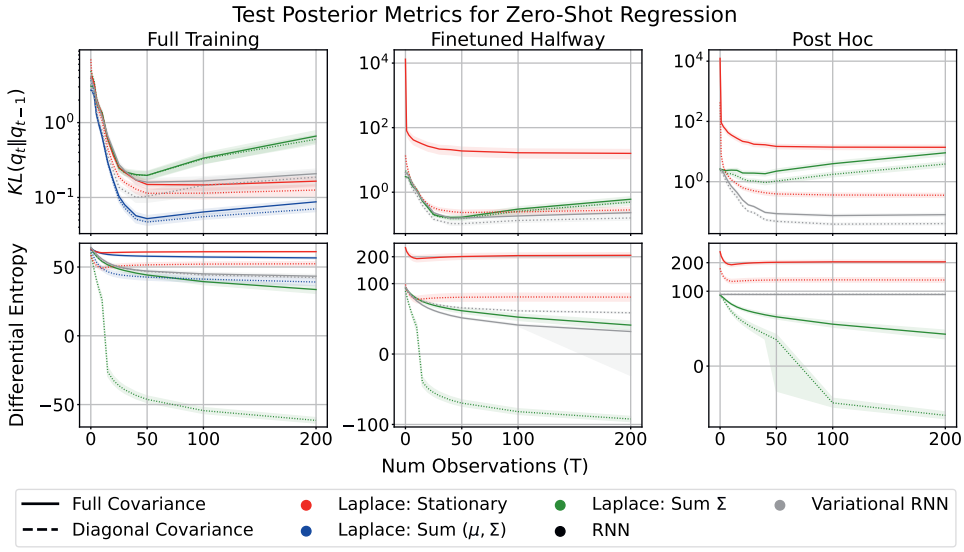


Figure 2.7: Posterior statistics during testing on the supervised task. The consecutive KL divergences (top) and entropy (bottom) should go down over time. The Laplace VRNN where we sum the covariances (green) is the only method that performs as expected for the entropy estimation but seems to grow more unstable for the KL-divergences. Results use $\beta = 10^{-2}$.

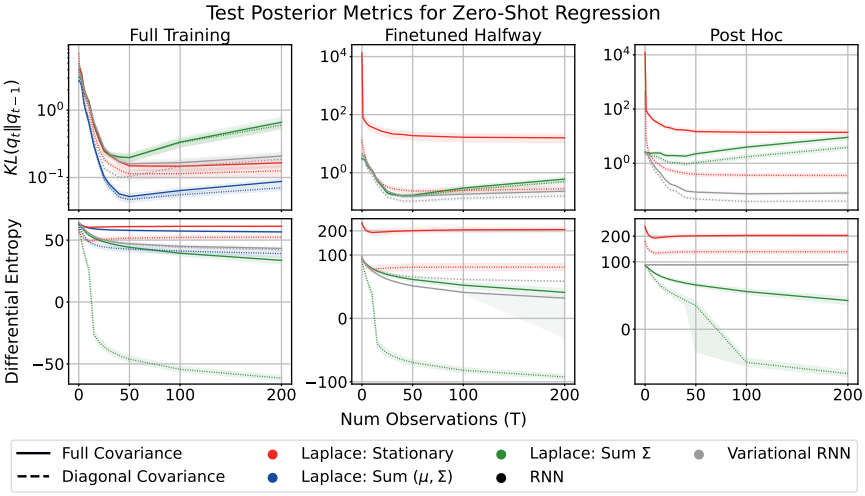


Figure 2.8: Complete plot of Figure 2.1 from the main paper to include the accumulation over means and covariances (blue) in the left plot. This was left out of the main paper to improve visibility. Results use $\beta = 10^{-2}$.

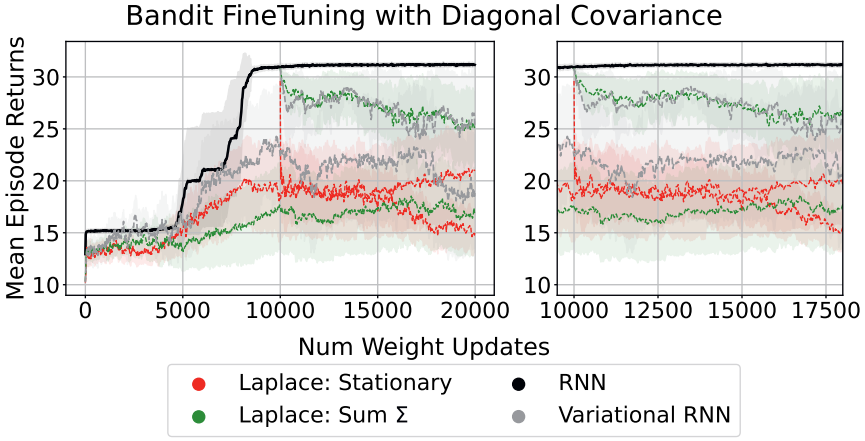


Figure 2.9: Zoom-in of Figure 2.3 for the bandit task when finetuning the deterministic model weights intermittently with a diagonal variational model. In this domain, it seems that finetuning slightly actually helps the expected training performance. Results use $\beta = 10^{-2}$.

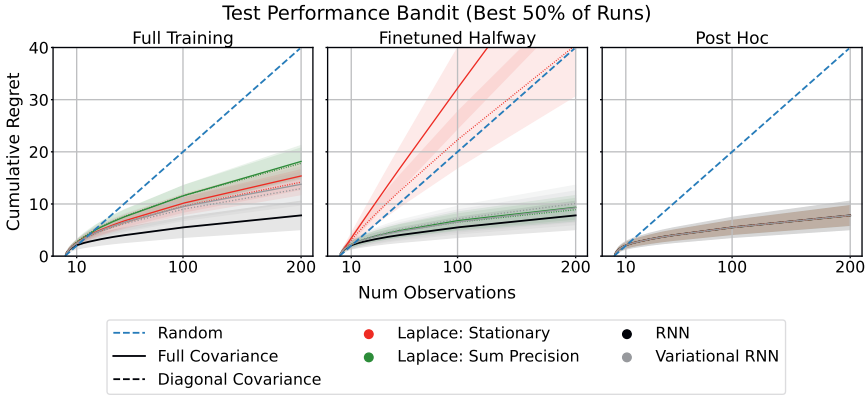


Figure 2.10: Cumulative regret of trained agents on the bandit task, lower is better. Since the training was quite unstable, about half of the repetitions did not find model weights with strong final performance. Only when we filtered out the better-than-median agents, did we find stronger than random performance. Results use $\beta = 10^{-2}$.

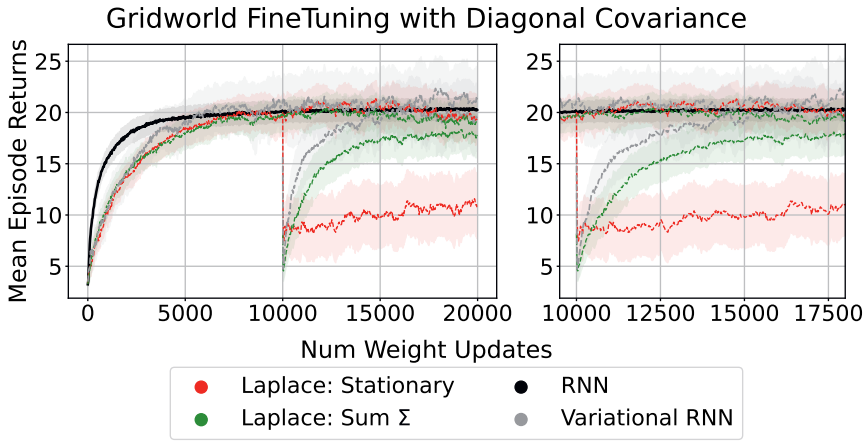


Figure 2.11: Zoom-in of Figure 2.3 for the gridworld task when finetuning the deterministic model weights intermittently with a diagonal variational model. In contrast to the bandit task, the agent needs to recover from this sudden change of additional model noise. Results use $\beta = 10^{-2}$.

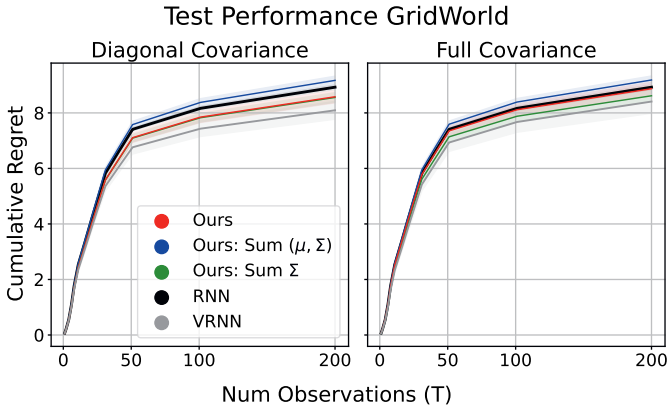


Figure 2.12: Cumulative regret of trained agents on the gridworld task, lower is better. All agents perform well on this task, the diagonal Variational RNN slightly outperforms all other agents.

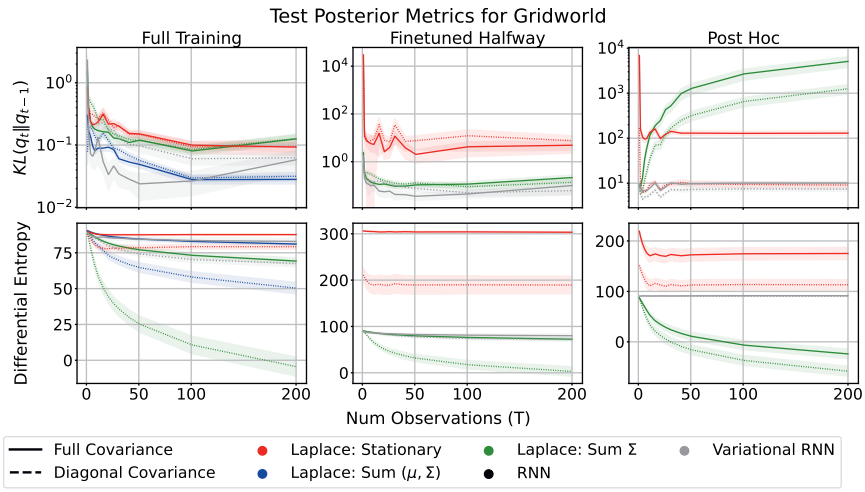


Figure 2.13: Complete plot of Figure 2.2 from the main paper to include the accumulation over mans and covariances (blue) in the left plot. Posterior statistics during testing on the gridworld task. The consecutive KL divergences (top) and entropy (bottom) should go down over time. Results use $\beta = 10^{-2}$.

3

Efficient Semi-Amortized Policy Inference

Seeking an improvement that makes a difference in the shorter term,
researchers seek to leverage their human knowledge of the domain,
but the only thing that matters in the long run is the leveraging of computation.

– Richard S. Sutton, in *The Bitter Lesson* (2019)

Optimal decision making over trajectories requires solving a typically intractable planning problem at each decision point. Reinforcement learning methods help in scaling up this procedure through learning, so that results of this expensive search problem can be stored in a cheaper function to be reused later. Existing literature in reinforcement learning often argues that a combination of amortized and online inference should yield more performant algorithms.

Towards this end, this chapter advances probabilistic planning algorithms for their use in model-based reinforcement learning. We first discuss background on recent Monte-Carlo tree search and sequential Monte-Carlo methods, and promote the latter method due to its greater scale-up potential to distributed compute resources. However, we also identify a variety of shortcomings of current sequential Monte-Carlo approaches for their use in reinforcement learning. In Section 3.3 we propose a number of improvements to the current state-of-the-art variational particle-filter planner based on these shortcomings, the subsequent Section 3.4 shows empirically that each contribution improves agent performance.

Our results show that particle-filter planners can match or surpass state-of-the-art Monte-Carlo tree search planners in terms of data-efficiency. Moreover, they can also improve training runtime of agents in a way that scales with additional planning budget. This is a serendipitous side effect of our setting where training happens to be bottlenecked by neural network learning.

The contents of this chapter have been published in (de Vries, He, Oren, & Spaan, 2025).

3.1 Introduction

Monte-Carlo tree search (MCTS) with neural networks (Browne et al., 2012) has enabled many recent successes in sequential decision making problems such as board games (Silver et al., 2018), Atari (Ye et al., 2021), and algorithm discovery (Fawzi et al., 2022a, Mankowitz et al., 2023a). These successes demonstrate that combining decision-time planning and reinforcement learning (RL) often outperforms methods that utilize search or deep neural networks in isolation. Naturally, this has stimulated many studies to understand the role of planning in combination with learning (Bertsekas, 2022, de Vries et al., 2021, Guez et al., 2019, Hamrick et al., 2021, He et al., 2024) along with studies to improve specific aspects of the base algorithm (Antonoglou et al., 2022, Danihelka et al., 2022, Hubert et al., 2021, Oren, Vadocz, et al., 2025).

Although MCTS has been a leading technology in recent breakthroughs, the tree search is inherently sequential, can deteriorate agent performance at small planning budgets (Grill et al., 2020), and requires significant modifications for general use in RL (Hubert et al., 2021). The sequential nature of MCTS is particularly crippling as it limits the full utilization of modern hardware such as GPUs. Despite follow-up work attempting to address specific issues, alternative planners have since been explored that avoid these flaws. Successful alternatives in this area are often inspired by stochastic control (Åström, 1970, Del Moral, 2004), examples include path integral control (Hansen et al., 2022, Theodorou et al., 2010, G. Williams et al., 2015) and related sequential Monte-Carlo (SMC) methods (Chopin & Papaspiliopoulos, 2020, Naesseth et al., 2019).

Specifically, recent variational SMC planners (Macfarlane et al., 2024, Naesseth et al., 2018) have shown great potential in terms of generality, performance, and scalability to parallel compute. These methods adopt a particle filter for trajectory smoothing to enable planning in RL (Piché et al., 2019). However, the distribution of interest for these particle filters do not perfectly align with learning and exploration for RL agents. Namely, recent SMC methods focus on estimation of the trajectory distribution under an unknown policy, and not the actual unknown policy at the state where we initiate the planner. We find that this mismatch can cause SMC planners to suffer from unnecessarily high-variance estimation and waste much of their compute and data during planning. In other words, online planning in RL should serve as a local *approximate policy improvement* (Chan et al., 2022, Sutton & Barto, 2018). Fortunately, existing MCTS and SMC literature provides various directions to achieve this (Danihelka et al., 2022, Grill et al., 2020, Lawson et al., 2018, Moral et al., 2010, Svensson et al., 2015), but their use in SMC-planning remains largely unrealized.

This paper aims to address the current limitations of bootstrapped particle filter planners for RL by drawing inspiration from MCTS. Our contributions are 1) to make more accurate estimates of statistics extracted by the planner, at the start of planning, and 2) enhancing data-efficiency inside the planner. We address the pervasive *path-degeneracy* problem in SMC by backing up accumulated reward and value data to perform policy inference and construct value targets. Next, to reduce the variance of estimated statistics due to resampling, we use exponential *twisting* functions to improve the sampling trajectories inside the planner. We also impose adaptive trust-region constraints to a prior policy, to control the bias-variance trade-off in sampling proposal trajectories. Finally, we modify

the resampling method (Naesseth et al., 2019) to correct particles that become permanently stuck in absorbing states due to termination in the baseline SMC. We dub our new method *Trust-Region Twisted* SMC and demonstrate improved sample-efficiency and runtime scaling over SMC and MCTS baselines in discrete and continuous domains.

3.2 Background

We want to find an optimal policy π for a sequential decision-making problem, which we formalize as an infinite-horizon Markov decision process (MDP) (Sutton & Barto, 2018). We define states $S \in \mathcal{S}$, actions $A \in \mathcal{A}$, and rewards $R \in \mathbb{R}$ as random variables, where we write $H_{1:T} = \{S_t, A_t\}_{t=1}^{T}$ as the joint random variable of a *finite* sequence,

$$p_\pi(H_{1:T}) = \prod_{t=1}^{T} \pi(A_t|S_t) p(S_t|S_{t-1}, A_{t-1}), \tag{3.1}$$

where $p(S_1|A_0, S_0) \triangleq p(S_1)$ is the initial state distribution, $p(S_{t+1}|S_t, A_t)$ is the transition model, and $\pi(A_t|S_t)$ is the policy. We denote the set of admissible policies as $\Pi \triangleq \{\pi | \pi : \mathcal{S} \to \mathbb{P}(\mathcal{A})\}$, our aim is to find a parametric $\pi_\theta \in \Pi$ (e.g., a neural network) such that $\mathbb{E}_{p_{\pi_\theta}(H_{1:T})}[\sum_{t=1}^{T} R_t]$ is maximized, where we abbreviate $R_t = R(S_t, A_t)$ for the rewards. For convenience, we subsume the discount factor $\gamma \in [0, 1]$ into the transition probabilities as a termination probability of $p_{\text{term}} = 1 - \gamma$ assuming that the MDP always ends up in an absorbing state S_T with zero rewards.

3.2.1 Control as Inference

The reinforcement learning problem can be recast as a probabilistic inference problem through the control as inference framework (Attias, 2003, Levine, 2018, Toussaint & Storkey, 2006). This reformulation has lead to successful algorithms like MPO (Abdolmaleki et al., 2018) that naturally allow *regularized* policy iteration (Geist et al., 2019), which is highly effective in practice with neural network approximation. Additionally, it enables us to directly use tools from Bayesian inference on our graphical model.

To formalize this, the distribution for $H_{1:T}$ can be conditioned on a binary outcome variable $\mathcal{O}_t \in \{0, 1\}$, then given a likelihood for $p(\mathcal{O}_{1:T} = 1|H_{1:T}) = p(\mathcal{O}_{1:T}|H_{1:T})$ this gives rise to a posterior distribution $p(H_{1:T}|\mathcal{O}_{1:T})$. Typically, we use the exponentiated sum of rewards for the (unnormalized) likelihood $p(\mathcal{O}_{1:T}|H_{1:T}) \propto \prod_{t=1}^{T} \exp R_t$.

Definition 3.1. The posterior factorizes as

$$p_\pi(H_{1:T}|\mathcal{O}_{1:T}) = \prod_{t=1}^{T} p_\pi(A_t|S_t, \mathcal{O}_{t:T}) p(S_t|S_{t-1}, A_{t-1}),$$

assuming that $S_{t+1} \perp\!\!\!\perp \mathcal{O}_{1:T}|S_t, A_t$ and $A_t \perp\!\!\!\perp \mathcal{O}_{<t}|S_t$.

The key part of Definition 3.1 is the posterior policy $p_\pi(A_t|S_t, \mathcal{O}_{t:T})$ which, using Bayes rule, reads as

$$p_\pi(A_t|S_t, \mathcal{O}_{t:T}) \propto \pi(A_t|S_t) \exp[\ln p(\mathcal{O}_{t:T}|S_t, A_t)]. \tag{3.2}$$

This connects us to an (expected-reward) maximum-entropy reinforcement learning setting (Toussaint, 2009, Ziebart, 2010), since the exponent admits a soft-Bellman recursion,

$$\ln p(\mathcal{O}_{t:T}|S_t, A_t) = R(S_t, A_t) + \ln \mathbb{E}\, e^{V^{\pi}_{soft}(S_{t+1})}, \tag{3.3}$$

where the expectation is over the dynamics $p(S_{t+1}|S_t, A_t)$. An important property of the posterior policy is that it is not an optimal policy in the traditional objective $\mathbb{E}_{p_\pi}[\sum_t R_t]$, but only provides an improvement over a prior policy π.

Theorem 3.1. *The posterior policy $p(A_t|S_t, \mathcal{O}_{t:T}), \forall t$, is the optimal policy q^* for the regularized MDP,*

$$q^* = \arg\max_{q \in \Pi} \mathbb{E}\left[\sum_{t=1}^{T} R_t - KL(q(a|S_t) \| \pi(a|S_t))\right],$$

where q^* guarantees a policy improvement in the unregularized MDP, $\mathbb{E}_{p_{q^*}(H_{1:T})}[\sum_{t=1}^{T} R_t] \geq \mathbb{E}_{p_\pi(H_{1:T})}[\sum_{t=1}^{T} R_t]$.

Proof. See Appendix 3.A.3, the original result by Rawlik et al. (2012), or Sec. 2.4 of (Levine, 2018). □

The optimal regularized policy q^* can be interpreted as a *variational* policy[1] that minimizes the evidence gap (Bishop, 2007), or conversely, maximizes the evidence lower-bound to $\ln p(\mathcal{O}_{1:T})$. In practice, this can be used in expectation-maximization (EM) methods (Neal & Hinton, 1998, Toussaint et al., 2008) to iteratively update the prior policy $\pi_{(n+1)} \leftarrow q^*_{(n)}$. This gives rise to an effective framework for approximate policy iteration that is both amenable to gradient based updating of π_θ and provably recovers the traditional (locally) optimal policy $\max_{\pi \in \Pi} \mathbb{E}_{p_\pi}[\sum_t R_t]$ as $n \to \infty$ (see Appendix 3.A).

3.2.2 Particle Filter Planning for RL

Although the estimation of the regularized optimal policy q^* mitigates a number of practical challenges in deep RL, it also requires re-estimating $q^*_{(n)}$ after each update to the prior $\pi_{(n+1)}$. Additionally, the posterior policy for any π always requires the solutions to the soft value-estimation problem for Q^{π}_{soft} and V^{π}_{soft}. Algorithms like MPO deal with this by approximating the value using neural networks $Q^{\pi}_{\theta} \approx Q^{\pi}_{soft}$ to then estimate the posterior policy through Monte-Carlo. Intuitively, this can be interpreted as a 1-timestep approximation to q^*. To see this, evaluating $Q^{\pi}_{\theta}(S_t, A_t)$ over samples $A_t \sim \pi(A_t|S_t)$ amortizes the message-passing process for evaluating Q^{π}_{soft} into a direct (cheap) mapping from $\mathcal{S} \times \mathcal{A} \to \mathbb{R}$. The main limitation of this approach is the bias induced by this new function Q^{π}_{θ}. Towards this end, sequential Monte-Carlo (SMC) methods (Naesseth et al., 2019, Piché et al., 2019) offer a powerful model-based strategy for improving the estimate of q^* through a multi-timestep.

[1] From this point on we will interchange the regularized optimal policy $q^*_{(n)}(A_t|S_t)$ with the posterior policy $p_{\pi_{(n)}}(A_t|S_t, \mathcal{O}_{t:T})$.

The SMC algorithm for RL (Piché et al., 2019) is a sequential importance sampling (IS) method that aims to draw samples $H_{1:t}$ from the posterior through a proposal distribution $p_q(H_{1:t})$ (as in Eq. 3.1 for some $q \in \Pi$). By definition, our graphical model (Def. 3.1) factorizes recursively,

$$p_\pi(H_{1:T}|\mathcal{O}_{1:T}) = p_\pi(H_{1:t}|\mathcal{O}_{1:T}) \cdot p_\pi(H_{t+1:T}|H_t, \mathcal{O}_{t+1:T}), \tag{3.4}$$

meaning that we can sample data from $H_{1:t} \sim p_q(H_{1:t})$ and accumulate the IS-weights sequentially.

Corollary 3.2 (Sequential importance sampling)**.** *Assuming access to the transition model $p(S_{t+1}|S_t, A_t)$, we obtain the importance sampling weights for $p_\pi(H_{1:t}|\mathcal{O}_{1:T})/p_q(H_{1:t})$,*

$$w_t = w_{t-1} \cdot \frac{\pi(A_t|S_t)}{q(A_t|S_t)} \exp(R_t) \frac{\mathbb{E}[\exp V^\pi_{soft}(S_{t+1})]}{\exp V^\pi_{soft}(S_t)}.$$

Proof. The dynamics terms in the weights cancel out, the rest follows by definition from Equations 3.2, 3.3, and 3.4. □

In practice we cannot realistically compute w_t since it requires the soft-values V^π_{soft}. However, we can again approximate this $\widetilde{w}_t \approx w_t$ with the amortized estimate $V^\pi_\theta \approx V^\pi_{soft}$ at every intermediate timestep (Lawson et al., 2018, Pitt & Shephard, 1999). With this, we can estimate posterior statistics using a single forward pass through: sampling traces $H^{(i)}_{1:t}$, accumulating their approximate weights $\widetilde{w}^{(i)}_t$, and then normalizing these $\overline{w}^{(i)}_t = \frac{\widetilde{w}^{(i)}_t}{\sum_j \widetilde{w}^{(j)}_t}$ to obtain

$$\begin{aligned} \mathbb{E}_{p_\pi(H_{1:t}|\mathcal{O})} f(H_{1:t}) &= \mathbb{E}_{p_q}[w_t \cdot f(H_{1:t})] \\ &\approx \sum_{i=1}^{K} \overline{w}^{(i)}_t f(H^{(i)}_{1:t}), \quad H^{(i)}_{1:t} \sim p_q, \end{aligned} \tag{3.5}$$

where $f : \mathcal{H}^t \to \mathcal{Y}$ is some arbitrary function over the data. For instance, the statistic $f(H_{1:t}) = \sum_{j=1}^t R_j$ would yield an estimate of the expected finite-horizon sum of rewards.

Bootstrapped Filter The bootstrapped particle filter for RL, as proposed by Piché et al. (2019), improves the estimation in Eq. 3.5 through (periodic) *resampling*. This mitigates the issue of weight-impoverishment in sequential-IS, where some normalized weights dominate others $\overline{w}^{(i)}_t \gg \overline{w}^{(j)}_t, j \neq i$. A common strategy for this is multinomial resampling (Chopin & Papaspiliopoulos, 2020), as shown in Algorithm 1. This method samples a number of traces $H^{(i)}_{1:m} \sim p_q, i \in [1, K]$, referred to as particles, and periodically drops or duplicates these samples based on their weights $\overline{w}^{(i)}_{1:m}, m \in [1, t]$, before resetting their weights back to a uniform distribution (bootstrap). Prior work (Macfarlane et al., 2024, Piché et al., 2019) then estimates the policy $\hat{q}^*$ as a weighted mixture of point-masses,

$$\hat{q}^*(A_t = a|S_t) = \sum_{i=1}^{K} \overline{w}^{(i)}_{t+m} \delta(A_t^{(J^{(i)}_{t+m})} = a), \tag{3.6}$$

Algorithm 1 Bootstrapped Particle Filter for RL

Require: K (number of particles), m (depth)

Initialize:
- Ancestor identifier $\{J_1^{(i)} = i\}_{i=1}^K$
- States $\{S_1^{(i)} \sim p(S_1)\}_{i=1}^K$
- Weights $\{\widetilde{w}_0^{(i)} = 1\}_{i=1}^K$

for $t = 1$ to m **do**

// Update particles

$\{A_t^{(i)} \sim q(A_t|S_t^{(i)})\}_{i=1}^K$

$\{S_{t+1}^{(i)} \sim p(S_{t+1}|S_t^{(i)}, A_t^{(i)})\}_{i=1}^K$

$\{\widetilde{w}_t^{(i)} = \widetilde{w}_{t-1}^{(i)} \frac{\pi(A_t^{(i)}|S_t^{(i)})}{q(A_t^{(i)}|S_t^{(i)})} e^{R_t^{(i)}} \frac{\mathrm{E} \exp V_\theta^\pi(S_{t+1}^{(i)})}{\exp V_\theta^\pi(S_t^{(i)})}\}_{i=1}^K$

// Bootstrap (periodically) through resampling

$\{(J_t^{(i)}, S_{t+1}^{(i)}, A_t^{(i)})\}_{i=1}^K \sim \text{Multinomial}(K, \overline{w}_t)$

$\{\widetilde{w}_t^{(i)} = 1\}_{i=1}^K$

end for

return $\{J_{1:m}^{(i)}, H_{1:m}^{(i)}, \widetilde{w}_{1:m}^{(i)}\}_{i=1}^K$

where $J_{t+m}^{(i)} \in [1, K]$ is the index tracking each samples' ancestor at the start of planning t and $\delta(\cdot)$ is a Dirac delta function over $\mathcal{A}$ for the ancestor action-particles.

A key property of Algorithm 1 is that it can estimate $\hat{q}^*$ through a single forward pass of K particles from t to $t+m$ (Moral et al., 2010). This allows for parallel sampling and updating of particles, with communication only required during the resampling step. As a result, it scales efficiently on modern GPU hardware, with memory complexity of $\mathcal{O}(K)$ and a parallelized time complexity of $\mathcal{O}(m)$.

Variational SMC Intuitively, resampling improves our estimate of Eq. 3.5 by 'correcting' particles with low-likelihood to higher likelihood regions of the target distribution. However, resampling also introduces noise and would ideally not be needed (or to a lesser extent) if our proposal distribution p_q produced samples that better match our target. To this end, variational SMC methods (S. Gu et al., 2015, Naesseth et al., 2018) learn a proposal distribution q_θ from sample estimates of the posterior target $\mathbb{E}[\hat{q}^*] = q^*$. Macfarlane et al. (2024) use this strategy with neural networks to learn $q_\theta \approx q^*$. Given the neural network iterates $\{\theta_{(n)}\}_{n=1}^N$, their method also updates the prior policy $\pi_{(n+1)} \leftarrow q_{\theta_{(n)}}$ at every step n. In other words, they combine variational SMC in an EM-loop (Neal & Hinton, 1998) by using the learned proposals $q_{\theta_{(n)}}$ as the prior to regularize the posterior $p_{\pi_{(n+1)}}(A_t|S_t, \mathcal{O}_{t:T})$.

3.3 Tailoring Particle Filter Planning to RL

A key benefit of online planning in reinforcement learning (RL) is generating locally improved policies at each timestep t compared to a prior policy, which enhances learning targets and diversity of data (Hamrick et al., 2021, Silver et al., 2018). Despite recent progress in improving sequential Monte-Carlo (SMC) methods for local approximate *policy improvement* (Chan et al., 2022), persisting design choices from conventional particle

filtering do not align directly with this goal. For instance, we find that the problem of *path degeneracy*, inherent to particle filtering (Svensson et al., 2015), can cause policy inference in Eq. 3.6 to degenerate with deep search and waste much of the planner's generated data. We also argue that variational SMC planners still make inefficient use of their particle budget due to periodic resampling, which can cause particles to get stuck in terminal states and delays using value information. By contrasting this with Monte-Carlo tree search (MCTS), which specifically focuses on improving the policy at the root (Grill et al., 2020), we tailor the SMC-planner for local *policy improvement*. This preserves the forward-only implementation of particle filtering while incorporating benefits inspired by MCTS.

Thus, in this section we develop our method in three steps: we first adapt the proposal distribution to improve sample efficiency (Section 3.3.1), we then introduce revived resampling to deal efficiently with terminal states (Section 3.3.2), and finally we mitigate path degeneracy in the policy inference and the value targets (Sections 3.3.3 and 3.3.4). The complete pseudo-code, extending Algorithm 1 with each of these components, is provided in Appendix 3.C.

3.3.1 Adapting the Proposals for Sample-Efficiency

The resampling step in SMC (line 6, Alg. 1) is essential for redistributing particles that are unlikely under the posterior policy. Simultaneously, variational SMC methods (Macfarlane et al., 2024) can shift some of this responsibility to a learned proposal distribution $q_\theta \approx \hat{q}^*$ by generating better samples. This is useful as it reduces variance induced by resampling and amortizes posterior inference (Naesseth et al., 2018). However, in EM-loop style algorithms, the learned proposal distribution q_θ is an estimate of the posterior policy that is regularized to a prior from a previous iteration $q_\theta \approx p_{\pi_{(n-1)}}(A_t|S_t, \mathcal{O}_{t:T})$, and not to the current iteration $\pi_{(n)}$. This is computationally wasteful with infrequent resampling, as it can take multiple transitions before moving particles to rewarding trajectories (Lioutas et al., 2023).

For this reason, we extend the proposal distribution to more accurately reflect the *next* posterior policy $p_{\pi_{(n)}}(A_t|S_t, \mathcal{O}_{t:T})$ by using both a learned policy $q_\theta \equiv \pi_{(n)}$ and value $\exp Q^\pi_\theta$. This is also known as exponential twisting (Asmussen & Glynn, 2007) of the proposal, which is similarly used in the target distribution (Corollary 3.2). Twisting reduces the variance for $\hat{q}^*$ at the cost of some bias due to Q^π_θ. This enables lower particle budgets through improved particle efficiency, but also introduces the difficulty in controlling this trade-off. In this regard, MCTS-based methods can offer some insight for dealing with this.

Since AlphaZero (Silver et al., 2018), MCTS-based methods also often use a trained policy π_θ (conflated with the name prior distribution) to guide the search in combination with a P-UCT algorithm (Rosin, 2011). Crucially, the initial iterations of the algorithm rely more on π_θ, which is interpolated to a greedy policy over the estimated values Q^π in later iterations. Grill et al. (2020) show how this causes inferred policies from MCTS to track a regularized objective similarly to Theorem 3.1 (with an added *greediness* parameter) over consecutive iterations. This is relevant to our work, because it links the estimated quantity of our SMC planner to the approach taken by MCTS. The main difference persists in that the iterations of MCTS induce an adaptive regularizer through a proxy for value-accuracy.

Inspired by this, we formulate our proposal distribution through a constrained opti-

mization problem with an adaptive trust-region parameter $\epsilon_\alpha \in \mathbb{R}_{\geq 0}$. At each state S_t, the proposal q solves for a locally constrained program,

$$\begin{aligned} \max_{q \in \mathbb{P}(\mathcal{A}|S_t)} \quad & \mathbb{E}_{q(A_t|S_t)} Q_\theta^\pi(S_t, A_t), \\ \text{s.t.,} \quad & KL(q(a|S_t) \| \pi_\theta(a|S_t)) \leq \epsilon_\alpha, \end{aligned} \tag{3.7}$$

where $\alpha \in [0, 1]$ is a greediness tolerance level. The actual trust-region ϵ_α is then sandwiched between the prior π_θ and the greedy policy π^* over Q_θ^π, such that, $\epsilon_\alpha = \alpha \cdot KL(\pi^* \| \pi_\theta)$. In similar spirit to MCTS, this guarantees that SMC always searches trajectories that interpolate between maximizing Q_θ^π or sticking to the prior π_θ. The Lagrangian of this program is similar to Theorem 3.1 with the sum of rewards replaced by Q_θ^π and the KL term scaled by a temperature. Furthermore, Eq. 3.7 requires $KL(q \| \pi) \leq \epsilon$ at every $S_t \in \mathcal{S}$ instead of in expectation over $p_q(H)$. For solving this program, we use a bisection search using bootstrapped atoms from π_θ, which is both general and computationally cheap (see Appendix 3.B.5 for details).

3.3.2 Handling Terminal States with Revived Resampling

Although the control as inference framework from Section 3.2.1 enables the use of SMC methods for policy inference, it also introduces non-trivial caveats. In particular, the infinite horizon formulation in Eq. 3.1 can lead to wasting compute when handling terminal states as absorbing states in a forward-only SMC algorithm. Absorbing states result in transitions that loop back to the same state with zero reward, effectively treating this as part of the environment dynamics. In an SMC planner, this can cause some of the K particles to become trapped in these absorbing states. Although resampling should correct for this, reward sparsity and errors in value predictions $V_\theta^\pi \approx V_{soft}^\pi$ can render the weights to become nearly uniform, making these trapped particles indistinguishable from non-trapped ones. This is a problem, because trapped particles do not contribute to gathering information about future values.

Furthermore, the resampling step can also move particles *towards* absorbing states due to rewarding trajectories in previous steps. Consecutive resampling in the SMC planner, however, is then likely to move these particles away from these states again because they no longer accumulate reward. This phenomena is mostly problematic when performing policy inference according to Eq. 3.6 and is a consequence of the *path degeneracy* problem (Svensson et al., 2015).

To address the trapped particles, we leverage the strategy of MCTS to reset trajectories to earlier states within the search tree. In MCTS, each iteration of the algorithm completely resets the agent to the root state, enabling the accumulation of new trajectory and value information. However, this full reset limits the ability to explore deep inside the search tree, which is a key benefit of SMC with its depth parameter m. Therefore, we propose a 'revived resampling' strategy to move particles back to their last non-terminal state. This only requires caching an additional reference state for each particle, assuming that the environment correctly flags these as non-terminal. Resampling is then performed to these reference states instead of the current states (see Appendix 3.C).

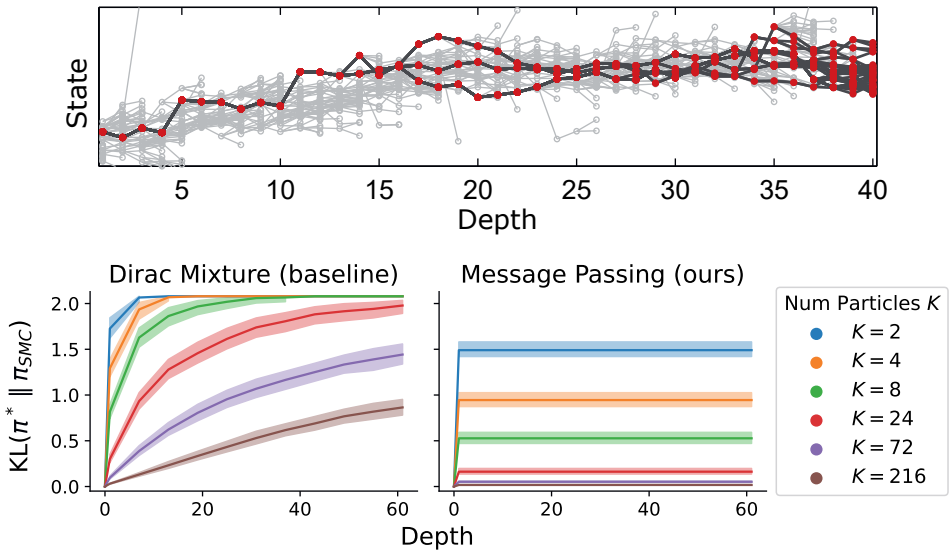


Figure 3.1: Illustration of *path degeneracy* in particle filters (top; adapted from Svensson et al., 2015) and the consequence on divergence from the optimal policy π^* (lower is better). With a finite budget of particles and *resampling*, deep search will concentrate the Dirac mixture of remaining ancestors to a single atom (bottom-left). We improve this through approximate message-passing to the ancestors, which does not degenerate the policy (bottom-right).

3.3.3 Mitigating Path Degeneracy for Policy Inference

A common problem in particle filtering is *path degeneracy*, where most particles collapse to a single ancestor due to resampling. This is illustrated in the top of Figure 3.1, the grayed-out trajectories highlight the discarded data due to resampling in typical forward-only SMC methods. This loss of ancestor diversity leads to a worse approximation of the distribution over states under our posterior policy (Svensson et al., 2015). We remark that in RL the consequence of this problem is exacerbated since we predominantly care about the root-ancestors. For this reason, SMC planners (Lioutas et al., 2023, Macfarlane et al., 2024, Piché et al., 2019) that perform policy inference for $\hat{q}^*$ using mixtures of point-masses from Eq. 3.6 show deteriorating approximation quality as depth increases, as shown in Figure 3.1 (bottom-left).

In contrast, MCTS does not suffer from path degeneracy because it does not discard any data. Policy inference is also typically done in MCTS by tracking a normalized state-action visitation counter that is updated in each iteration of the algorithm (Browne et al., 2012). Importantly, the normalized visit-counts can be used as both a behavior policy and learning target (Silver et al., 2018). Recent work however, has shown that using such a visitation-count policy can degrade performance when using low planning budgets. Instead, Danihelka et al. (2022) show that inferring a regularized policy (Grill et al., 2020) using the value statistics from search for the root state-actions avoids this problem.

However, SMC planners do not track visit-counts or perform backpropagation to accumulate value statistics. Since the importance sampling weights factorize recursively,

we can adopt the approach by Moral et al. (2010) to perform an online estimation to $\hat{Q}_{SMC} \approx Q^{\pi}_{soft}$ inside SMC using Eq. 3.3, to then construct a policy estimate similarly to Danihelka et al. (2022). At each SMC step t, we accumulate a current estimate of $\hat{Q}^{(j)}_{SMC}$ for any root-ancestor j as,

$$\hat{Q}^{(j)}_t = \hat{Q}^{(j)}_{t-1} + \begin{cases} 0, & \mathcal{J}^{(j)}_t = \emptyset, \\ \ln\left(\frac{1}{|\mathcal{J}^{(j)}_t|}\sum_{i\in\mathcal{J}^{(j)}_t}\frac{\widetilde{w}^{(i)}_t}{\widetilde{w}^{(i)}_{t-1}}\right), & \mathcal{J}^{(j)}_t \neq \emptyset, \end{cases}$$

where $\mathcal{J}^{(j)}_t = \{i \in [1,K] \mid J^{(i)}_t = j\}$ and $\hat{Q}^{(j)}_0 = 0$. In essence, this expression accumulates an average log-probability for the remaining particles belonging to ancestor j. We then use the values $\hat{Q}^{(j)}_{t+m} \approx Q^{\pi}_{soft}(S_t, A^{(j)}_t)$ to infer $\hat{q}^*$ by bootstrapping the atoms proportionally to $\pi_\theta(A^{(i)}_t|S_t)\exp\hat{Q}^{(i)}_{SMC}$. Figure 3.1 (bottom-right) shows that this eliminates policy deterioration as a function of depth, reducing the consequence of path degeneracy.

3.3.4 Search-Based Values for Learning Targets

The approximate message passing from the previous subsection mitigates the path degeneracy problem by utilizing all SMC generated data for policy inference. In essence, we are constructing a better estimate of the soft-value functions $V^{\pi}_{\theta} \approx V^{\pi}_{soft}$ (Lawson et al., 2018, Piché et al., 2019) to then estimate $\hat{q}^*$ as given by Eq. 3.2. The value functions themselves are then trained using some temporal difference (TD) method in an outer learning loop, e.g., using n-step returns (Sutton & Barto, 2018) given environment interactions outside of the planner (see also Appendix 3.B.3). So far, most prior work constructs these TD-learning targets by value-bootstrapping from one-step predictions given states in the generated datasets (Lioutas et al., 2023, Macfarlane et al., 2024, Piché et al., 2019). For instance, given a transition (S_t, A_t, R_t, S_{t+1}), a 1-step TD target would be computed as $Y_t = R_t + V^{\pi}_{\theta}(S_{t+1})$. However, this approach neglects most data generated by the planner by only considering the 1-step value at the next states S_{t+1}.

Similarly to recent versions of MuZero (Danihelka et al., 2022, Schrittwieser et al., 2020, Ye et al., 2021), we instead use the values estimated by the search algorithm $\hat{V}_{t+1}$ over $V^{\pi}_{\theta}(S_{t+1})$ to compute these outer TD-targets. Not only does this exploit the planner data, $\hat{V}_t$ is also objectively a better value estimator to use for policy improvement (i.e., within our EM-loop). Namely, the predictions by V^{π}_{θ} give the value of a previous posterior policy (off-policy), whereas $\hat{V}_t$ is an estimate of the current posterior policy (on-policy). During our testing, we found at low particle budgets for the SMC planner that it is important to control the variance of the inner value estimation. For this reason, we compute $\hat{V}_t$ for value-learning using the Retrace(λ) returns (Munos et al., 2016) instead of the importance-weighted (soft) Monte-Carlo estimate used for the policy. This only requires tracking a secondary statistic along with $\hat{Q}^{(j)}$.

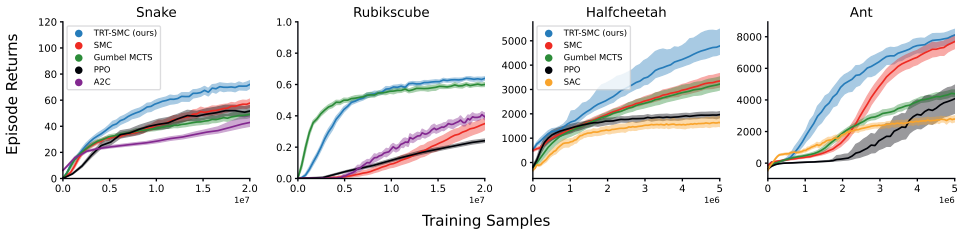


Figure 3.2: Per environment evaluation curves for a planning budget of $N = K \times m = 16$. Shaded regions give 99% two-sided BCa-bootstrap intervals over 30 seeds. This plot shows improved sample-efficiency of our method when a highly accurate simulator is available.

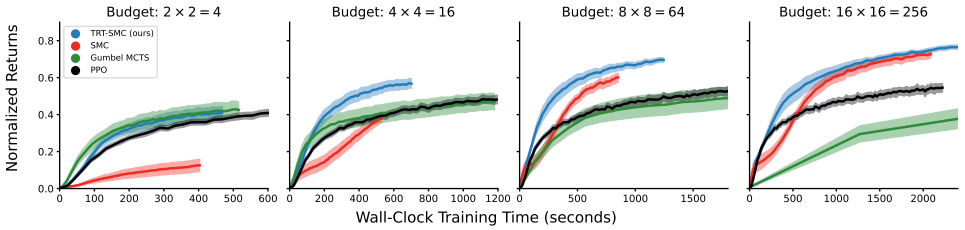


Figure 3.3: Normalized average curves for the discrete environments over increasing planning budgets to compare performance to *runtime*. Shaded regions give 99% two-sided BCa-bootstrap intervals over 2 times 30 seeds. Runtime was estimated by multiplying the training step with an interquartile mean of the runtime-per-step (see Appendix 3.B.2 for details and Appendix 3.D for sample-efficiency).

3.4 Experiments

We introduce our new method, *Trust-Region Twisted SMC* (TRT-SMC), a variational SMC method that is tailored for planning in RL. We claim that TRT-SMC implements a stronger approximate *policy improvement* over baseline approaches when used in an expectation-maximization framework (Abdolmaleki et al., 2018, Chan et al., 2022). The contribution of a stronger policy improvement operator can be isolated to 1) enhanced action-selection during training, and 2) improved learning targets (Hamrick et al., 2021). Therefore, we expect our method to yield higher final test returns and steeper learning curves in terms of sample-efficiency (training samples) and runtime efficiency (wallclock time). We compared our TRT-SMC against the variational SMC method by Macfarlane et al. (2024), and the current strongest Monte-Carlo tree search (MCTS) method, Gumbel AlphaZero by Danihelka et al. (2022).

We performed experiments in the Brax continuous control tasks (Freeman et al., 2021) and Jumanji discrete environments (Bonnet et al., 2024), using the authors' A2C and SAC results as baselines alongside our PPO implementation (Schulman et al., 2017). Although we compare sample-efficiency to the model-free baselines, this is only for reference since we do not account for the additional observed transitions by the planner in the main results. In other words, we are testing the setting where we assume access to a highly accurate simulator for planning purposes (see Appendix 3.D for additional results that also count the simulator samples). Performance is reported as the average offline return over 128 episodes. Unless otherwise stated, all experiments were repeated (retrained) across 30 seeds, with 99% two-sided BCa-bootstrap confidence intervals (Efron, 1987). For more

details, see Appendix 3.B.

3.4.1 Main Results

We show the evaluation curves for comparing sample-efficiency in Figure 3.2, where the planner-based methods used a budget of $N = 16$ transitions. For simplicity, we kept the depth m of the SMC planner uniform to the number of particles K, such that $K = m = \sqrt{N}$. Our ablations in the subsection 3.4.2 also show that keeping m and K somewhat in tandem is ideal for SMC. On each individual environment we observe that *our TRT-SMC shows steeper and higher evaluation curves than the baseline variational SMC method, which is in line with our expectations.*

Additionally, we compare the runtime scaling for additional planning budget in terms of normalized returns on the discrete Jumanji environments in Figure 3.3. We measured this by aggregating the average returns scaled by their environments' known min-max bounds. We see that with more planning budget that the baseline SMC often starts to approach our TRT-SMC, and that the Gumbel MCTS method scales poorly in wallclock time. Most importantly, these results show that *our TRT-SMC reliably scales in performance with additional budget, runtime, and training samples.*

3.4.2 Ablations

The main results show that our TRT-SMC method can perform better compared to the baseline model-free, variational SMC, and MCTS methods in terms of sample-efficiency and training runtime. However, we also want to asses: *to what extent do each of our separate contributions improve the base method?* Therefore, this section quantifies this across the different environments and parameter settings.

Proposal Distributions. Firstly, we assess the interplay of the proposal distribution with the planning budget through the aggregated final test performance on the Jumanji environments in Figure 3.4. This compares our TRT-SMC method when using either a twisted proposal distribution with a greediness tolerance of $\alpha = 0.1$, or $\alpha = 0$ (which only uses the prior π_θ). Similarly to the main results, at low particle budgets we observe significantly improved performance with the trust-region twisted proposal, but this gap decreases for $\alpha = 0$ at higher particle budgets. It also shows that performance scales favorably when the particle budget and the planner depth are in tandem with each other. *This performance improvement at low particle budgets matches our aim of increasing particle-efficiency.*

Secondly, we evaluated our TRT-SMC method varying the $\alpha \in [0, 1]$ parameter and uniformly scaling up the budget and depth (as done in Figure 3.3). We aggregated the normalized final test results over all tested environments as given by the sensitivity plot in Figure 3.5. Although we show the aggregated results here, we found that this pattern highly depends on the specific environment, we show the individual results in Appendix 3.D. In essence, these results show that *mixing between the prior π_θ and maximizing the predicted state-action value Q^π_θ improves performance compared to completely relying on either of them.*

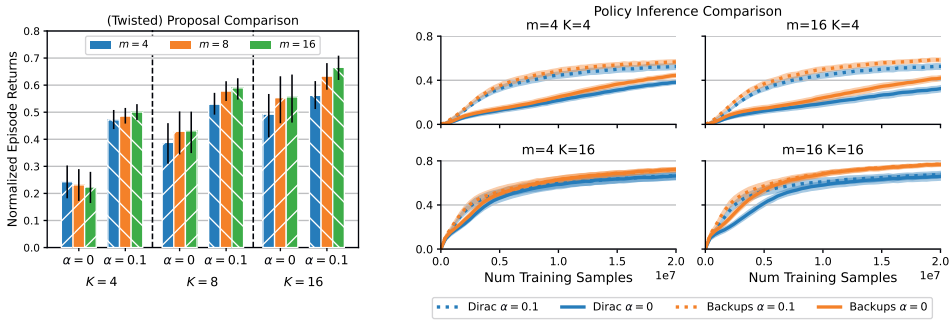


Figure 3.4: Final expected performance (left) and evaluation curves over training (right), for different proposal trust-region levels and policy inference methods. The left barplot only uses our message-passing method (backups) for estimating $\hat{q}^*$, whereas the right plot compares both the Dirac mixture and our method. Both plots show that the constrained proposals $\alpha = 0.1$ improve performance over the prior proposal $\alpha = 0$ at low particle budgets. The right plot also shows that our backup method for $\hat{q}^*$ does not degenerate with deep planning.

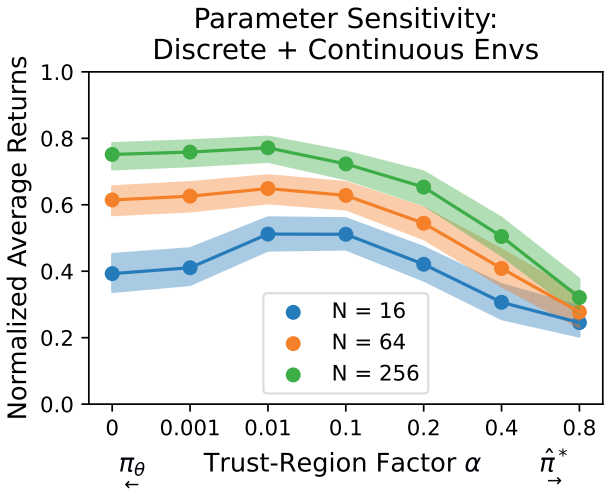


Figure 3.5: Sensitivity plot for the adaptive trust-region parameter α over increasing planning budgets $N = K \cdot m$ (where $K = m$). The y-axis indicates the normalized final test performance, aggregated across all tested environments. The pattern shows that at small planning budgets, a proposal distribution which accounts for the predicted Q^π_θ performs marginally better.

Policy Inference. We compare our method for policy inference to that of Eq. 3.6 by adopting a similar experiment setup as we used for the proposal ablations and varying these two choices. We report the results for this experiment in terms of their evaluation curves in the right of Figure 3.4 to also observe learning stability. We find that the final performance of the Dirac policy at $K = 4$ and $\alpha = 0$ shows decreased final performance when the planner depth is increased from $m = 4$ to $m = 16$. As expected, this effect does not occur for our message-passing method, although the proposal twisting seems to diminish this effect also. Most importantly, *our approach for computing $\hat{q}^*$ demonstrates a monotone*

Table 3.1: Confidence intervals for the final expected episode returns on Brax Ant for different value-estimation methods (top) and Jumanji Snake for the two resampling strategies (bottom).

Ablation Value	$\hat{\mu}$	$\hat{q}_{\alpha/2} - \hat{q}_{1-\alpha/2}$
Value Targets on Ant		
V_θ	6911.8	$6669.7 - 7146.7$
$\frac{1}{2}V_\theta + \frac{1}{2}V_{SMC}$	7214.7	$6973.9 - 7455.6$
V_{SMC}	**7457.1**	$7200.3 - 7715.3$
Resampling on Snake		
Baseline	44.7	$43.7 - 45.7$
Revived	45.7	$44.5 - 47.7$

improvement.

Value Targets. To compare the improvement by the search-based value targets, we tested on the Ant environment for experiment variety. We compared three variations of our TRT-SMC for constructing the *outer* learning targets by linearly interpolating the 1-step V_θ and the SMC-based value estimate V_{SMC}, such that $\hat{V} = \sigma \cdot V_\theta + (1-\sigma) \cdot V_{SMC}$, where we used $\sigma \in \{0, \frac{1}{2}, 1\}$. We aggregated the final mean performances across different planner depths $m \in \{4, 8\}$, particle budgets $K \in \{4, 8\}$, resampling periods $r \in \{1, 3\}$, and whether to use our revived resampling or not. To reduce moving parts, we did not use a twisted proposal ($\epsilon = 0$). In total, across 30 repetitions, this gave us 480 experiments for which we report final marginal performance in the top of Table 3.1. Although the intervals overlap slightly, *there is a trend that favors using SMC data for the learning targets.*

Revived Resampling. We evaluated the effect of using the revived particle resampling in a similar, experiment setup to the value-targets ablations. We tested on Jumanji Snake due to its sparse rewards and high likelihood of encountering terminal states (see Appendix 3.B.1). Interestingly, the marginal test results in the bottom of Table 3.1 show that *the revived resampling does not significantly differ from the baseline.*

3.5 Related Work

The connection of reinforcement learning (RL) to the statistical estimation of a probabilistic graphical model (Levine, 2018) has in recent years proven useful in borrowing tools from Bayesian estimation for optimal policy inference. Although we focus on sequential Monte-Carlo (SMC) methods (Hoffman et al., 2007, Piché et al., 2019), this connection has also been used to exploit stochastic control methods like TD-MPC (Hansen et al., 2024, Theodorou et al., 2010). Similarly, prior work has partially explored some of the modifications that we

make to SMC-planners, either in isolation or in different contexts. This paper therefore reinforces the connection between probabilistic inference and RL, but also links it back to recent approaches in Monte-Carlo tree search (MCTS) (Browne et al., 2012, Schrittwieser et al., 2020, Silver et al., 2018, S. Wang et al., 2024).

As discussed in Section 3.2.1, our method builds on the variational SMC approach by Macfarlane et al. (2024). Similarly, they also utilize trust-region methods, but on the neural network parameters during optimization and on the target distribution for SMC. Our setup is more comparable to recent MCTS methods (Danihelka et al., 2022, S. Wang et al., 2024) since we only impose trust-regions on the proposals and nothing else. In other words, we focus specifically on the *policy improvement* part of the algorithm. Then, Lioutas et al. (2023) also explore a type of 'twisted' proposals, they sample auxiliary action-particles and weight these with heuristic factors $\exp Q_\theta^\pi$ before computing transitions (Stuhlmülller et al., 2015). However, their approach has two issues: they mix the normalization of the auxiliary actions across particles and they do not impose sufficient regularization to trade-off the prior policy and the values.

Finally, our contributions are strongly tied to mitigating path degeneracy in SMC planners, which is a common theme in particle filtering (Chopin & Papaspiliopoulos, 2020). For instance, our estimation of the policy (and values) is similar to the online estimation of any time-separable function described by Moral et al. (2010), which was also motivated by path degeneracy. Although we did not consider it here, there are many promising directions for future work in this area, like using anchor particles (Svensson et al., 2015), adaptive resampling (Naesseth et al., 2019), or Rao-Blackwellisation (Casella & Robert, 1996, Danihelka et al., 2022).

3.6 Conclusion

This paper tailors a particle filter planner for its use within deep reinforcement learning. Specifically, we address default design choices within variational sequential Monte-Carlo that become problematic when applying these methods to perform policy inference. Our contributions take inspiration from recent Monte-Carlo tree search methods to mitigate the path degeneracy problem, make better use of the data generated by the planner, and to improve planning budget utilization. Experiments show that our *Trust-Region Twisted sequential Monte-Carlo* (TRT-SMC) scales favorably to varying planning budgets in terms of runtime and sample-efficiency over the baseline policy improvement methods. We hope that our approach inspires others to leverage more of the tools from both the planning and Bayesian inference literature, to further enhance the sample-efficiency and runtime properties of reinforcement learning algorithms.

Appendix

3.A Derivations

We first restate the factorization of the marginal in Eq. 3.1 from the main text,

$$p_\pi(H_{1:T}) = \prod_{t=1}^{T} \pi(A_t|S_t)p(S_t|S_{t-1}, A_{t-1}),$$

with $p(S_1|A_0, S_0) \triangleq p(S_1)$ being the initial state distribution, $p(S_{t+1}|S_t, A_t)$ is the transition model, and $\pi(A_t|S_t)$ is the policy. We denote the set of admissible policies as $\Pi \triangleq \{\pi|\pi : \mathcal{S} \to \mathbb{P}(\mathcal{A})\}$. We drop subscripts for $H_{1:T}$ if the indexing is clear from the text.

3.A.1 Lower-bound

For completeness, we show below that the factorization of $p_\pi(H_{1:T}|\mathcal{O}_{1:T} = 1)$, by Definition 3.1, recovers the lower-bound term for q^* shown in Theorem 3.1. This is done through the well-known decomposition of the log-likelihood on the marginal distribution for the outcome variable $\mathcal{O}_{1:T} \in \{0, 1\}^T$. See Section 3.2.1 for a description on the meaning of this variable, again we will abbreviate $\mathcal{O} = 1$ simply as $\mathcal{O}$.

Lemma 3.3 (Decomposition log-likelihood, c.f., Ch 9.4 of Bishop (2007))**.**

$$\ln p_\pi(\mathcal{O}) = \underbrace{\mathbb{E}_{p_q(H)}\left[\sum_{t=1}^{T} R_t - KL(q(a|S_t)\|\pi(a|S_t))\right]}_{\text{Evidence Lower-Bound}} + \underbrace{KL\left((p_q(H)\|p_\pi(H|\mathcal{O})\right)}_{\text{Evidence Gap}} \tag{3.8}$$

Proof. Assume an importance sampling distribution $q \in \Pi$ for $\pi \in \Pi$ such that it has sufficient support over H and $p_\pi(H) > 0 \implies p_q(H) > 0$ almost everywhere.

$$\begin{aligned}
\ln p_\pi(\mathcal{O}) &= \ln \frac{p_\pi(\mathcal{O}, H)}{p_\pi(H|\mathcal{O})} \\
&= \mathbb{E}_{p_q(H)}\left[\ln\left(\frac{p_\pi(\mathcal{O}, H)}{p_\pi(H|\mathcal{O})}\frac{p_q(H)}{p_q(H)}\right)\right] \\
&= \mathbb{E}_{p_q(H)} \ln \frac{p_\pi(\mathcal{O}, H)}{p_q(H)} + \mathbb{E}_{p_q(H)} \ln \frac{p_q(H)}{p_\pi(H|\mathcal{O})} \\
&= \mathbb{E}_{p_q(H)}\left[\ln p(\mathcal{O}|H) + \ln \frac{p_\pi(H)}{p_q(H)}\right] + \mathbb{E}_{p_q(H)} \ln \frac{p_q(H)}{p_\pi(H|\mathcal{O})} \\
&= \mathbb{E}_{p_q(H)}\left[\ln p(\mathcal{O}|H) - KL(p_q(H)\|\pi(H))\right] + KL(p_q(H)\|p_\pi(H|\mathcal{O})) \\
&= \mathbb{E}_{p_q(H)}\left[\sum_t R_t - KL(q(a|S_t)\|\pi(a|S_t))\right] + KL(p_q(H)\|p_\pi(H|\mathcal{O}))
\end{aligned}$$

where in the last step the transition terms for $p_\pi(H)$ and $p_q(H)$ cancel out. See the work by Levine (2018) for comparison. □

The result from Lemma 3.3 shows that the log-likelihood is decomposed into an evidence lower-bound and an evidence gap. This inequality becomes tight when q is simply set to the posterior policy $q^*(H) \equiv p_\pi(H|\mathcal{O})$. Thus, motivating the maximization objective for the lower-bound as given in Theorem 3.1, or equivalently, by minimizing the evidence gap as considered by Levine (2018).

3.A.2 Regularized Policy Improvement

The result below gives a brief sketch that the expectation-maximization loop (Neal & Hinton, 1998) generates a sequence of regularized Markov decision processes (MDPs) that eventually converges to a locally optimal policy. This result is a nice consequence of the control-as-inference framework. A more general discussion outside of the expectation-maximization framework can be found in the work by Geist et al. (2019).

Lemma 3.4 (Regularized Policy Improvement)**.** *The solution q^* to the problem,*

$$\max_{q \in \Pi} \mathbb{E}\left[\sum_{t=1}^{T} R_t - KL(q(a|S_t) \| \pi(a|S_t))\right],$$

guarantees a policy improvement in the unregularized MDP, $\mathbb{E}_{p_{q^*}(H)}[\sum_t R_t] \geq \mathbb{E}_{p_\pi(H)}[\sum_t R_t]$.

Proof. Lemma 3.3 shows that the solution q^* is equivalent to the posterior policy distribution $p_\pi(H_{1:T}|O_{1:T})$. This implies that for each state $s \in \mathcal{S}$, we have, $q^*(a|s) \propto \pi(a|s) \exp Q^\pi_{soft}(s,a)$. The exponential over $Q^\pi_{soft}(s,a)$ in the posterior policy q^* interpolates π to the greedy policy by shifting probability density to actions with larger expected cumulative reward. Thus, q^* provides a policy improvement over π. □

3.A.3 Proof of Theorem 3.1

Proof of Theorem 3.1. Lemma 3.3 shows how the posterior policy coincides with the optimal policy q^* in a regularized Markov decision process (MDP). Then Lemma 3.4 describes that this gives a policy improvement in the unregularized MDP. Iterating this process (e.g., an expectation-maximization loop) yields consecutive improvements to the prior $\pi_{(n)} \leftarrow q^*_{(n-1)}$ and guarantees a locally optimal π^* in the unregularized MDP as $n \to \infty$, which also implies $KL(q^*_{(n)} \| \pi_{(n-1)}) \to 0$. □

3.A.4 Importance sampling weights

For completeness, we give a detailed derivation for the result presented in Corollary 3.2. This derivation differs from the one given by Piché et al. (2019) in Appendix A.4. Our derivation corrects for the fact that we don't need to compute an expectation over the transition function for $\exp V^\pi_{soft}(S_t)$ in the denominator. However, this is only a practical difference (i.e., how the algorithm is implemented) to justify our calculation.

Corollary 3.5 (Restated Corollary 3.2)**.** *Assuming access to the transition model $p(S_{t+1}|S_t, A_t)$,*

we obtain the importance sampling weights for $p_\pi(H_{1:t}|\mathcal{O}_{1:T})/p_q(H_{1:t})$,

$$w_t = w_{t-1} \cdot \frac{\pi(A_t|S_t)}{q(A_t|S_t)} \exp(R_t) \frac{\mathbb{E}[\exp V^\pi_{soft}(S_{t+1})]}{\exp V^\pi_{soft}(S_t)},$$

Proof. For any statistic $f(\cdot)$, we have,

$$\begin{aligned}
\mathbb{E}_{p_\pi(H_{1:t}|\mathcal{O}_{1:T})} f(H_{1:t}) &= \mathbb{E}_{p_q(H_{1:t})}[w_t \cdot f(H_{1:t})] \\
&= \mathbb{E}_{p_q(H_{1:t})}\left[\frac{p_\pi(H_{1:t}|\mathcal{O}_{1:T})}{p_q(H_{1:t})} f(H_{1:t})\right] \\
&= \mathbb{E}_{p_q(H_{1:t})}\left[\frac{p_\pi(S_t, A_t|H_{<t}, \mathcal{O}_{1:T}) p_\pi(H_{<t}|\mathcal{O}_{1:T})}{p_q(S_t, A_t|H_{<t}) p_q(H_{<t}))} f(H_{1:t})\right] \\
&= \mathbb{E}_{p_q(H_{1:t})}\left[w_{t-1} \cdot \frac{p_\pi(S_t, A_t|S_{t-1}, A_{t-1}, \mathcal{O}_{t:T})}{p_q(S_t, A_t|S_{t-1}, A_{t-1})} \cdot f(H_{1:t})\right]
\end{aligned}$$

where the last step follows from the Markov property. Then, we get,

$$\begin{aligned}
\frac{p_\pi(S_t, A_t|S_{t-1}, A_{t-1}, \mathcal{O}_{t:T})}{p_q(S_t, A_t|S_{t-1}, A_{t-1})} &= \frac{p_\pi(A_t|S_t, \mathcal{O}_{t:T}) p(S_t|S_{t-1}, A_{t-1})}{q(A_t|S_t) p(S_t|S_{t-1}, A_{t-1})} = \frac{p_\pi(A_t|S_t, \mathcal{O}_{t:T})}{q(A_t|S_t)} \\
&= \frac{\pi(A_t|S_t)}{q(A_t|S_t)} \frac{p_\pi(\mathcal{O}_{t:T}|S_t, A_t)}{p_\pi(\mathcal{O}_{t:T}|S_t)} = \frac{\pi(A_t|S_t)}{q(A_t|S_t)} \frac{\exp Q^\pi_{soft}(S_t, A_t)}{\exp V^\pi_{soft}(S_t)} \\
&= \frac{\pi(A_t|S_t)}{q(A_t|S_t)} \exp(R_t) \frac{\mathbb{E}[\exp V^\pi_{soft}(S_{t+1})]}{\exp V^\pi_{soft}(S_t)}.
\end{aligned}$$

□

3.B Experiment Details

Our code can be found at https://github.com/joeryjoery/trtpi.

3.B.1 Environments

We used the Jumanji 1.0.1 implementations of the Snake-v1 and Rubikscube-partly-scrambled-v0 environments (Bonnet et al., 2024), code is available at https://github.com/instadeepai/jumanji. For the Brax 0.10.5 implementation we used the Ant and Halfcheetah environments using the 'spring' backend, code is available at https://github.com/google/brax.

Snake environment details:

- The observation is a 12x12 image with 5 channels that indicate the position of the fruit and positional features of the snake, it also gives the integer number of steps taken which we encode as a bit-vector for the neural network.

- The action space is a choice over 4 integers that indicate moving the snake: up, down, left, or right.
- The reward is zero everywhere, except for a +1 when the fruit is picked up.
- The agent terminates after 4000 steps or when colliding with itself.

Rubikscube environment details:

- The observation is a 3D integer tensor of shape $6 \times 3 \times 3$ with values in $[0, 1, 2, 3, 4, 5]$ indicating the color. It also gives the integer number of steps taken which we encode as a bit-vector for the neural network.
- The action space is a 3 dimensional integer array to choose the face to turn, depth of the turn, and direction of the turn. For a 3x3 cube this action-space induced 18 combinations.
- The reward is zero everywhere, except for a +1 when the puzzle is solved.
- The agent terminates after 20 steps or when solving the puzzle.
- We used the partly-scrambled version of this environment, which means that the solution is at most 7 actions removed from any of the starting states.

Brax environment details:

- The observations are continuous values with 27 dimensions for Ant and 18 dimensions for Halfcheetah.
- The action space is a bounded continuous vector between $[-1, 1]^D$, with $D = 8$ for Ant and $D = 6$ for Halfcheetah.
- The reward is a dense, but involved formula that penalizes energy expenditure (norm of the action) and rewards the agent for moving in space (in terms of spatial coordinates).
- The agent terminates after 4000 steps or when ending up in a unhealthy joint-configuration.

3.B.2 Hardware Requirements

All experiments were run on a GPU cluster with a mix of NVIDIA GeForce RTX 2080 TI 11GB, Tesla V100-SXM2 32GB, NVIDIA A40 48GB, and A100 80GB GPU cards. Each run (random seed/repetition) required only a few CPU cores (2 logical cores) with a low memory budget (e.g., 4GB). For our most expensive singular experiments we found that we needed about 6GB of VRAM at most, and that the replay buffer size is the most important parameter in this regard. Roughly speaking, we found that the SMC based agents all completed both training and evaluation under 30 minutes on Snake with a budget of $K = 8$ particles and a depth of $m = 8$, the Gumbel MCTS required 3 hours for a budget of $N = 64$.

Training Time Estimation. To estimate the training runtime in seconds (in Figure 3.3), we used an estimator of the the runtime-per-step and multiplied this by the current training iteration to obtain a cumulative estimate. For each training configuration, we measured the runtime-per-step and computed an interquartile mean over 1) the random seeds for relative wallcock time and 2) the training iterations themselves. This estimator should more robustly deal with the variations in hardware, the compute clusters' background load, and XLA dependent compilation. Of course, estimating runtime is strongly limited to the hardware and software implementation, and our results should only hint towards a trend of improved scaling to parallel compute for the planner algorithms.

3.B.3 Hyperparameters, Model Training, and Software Versioning

The hyperparameters for our experiments are summarized in the following tables:

- Table 3.3: Shared parameters across experiments.
- Table 3.4: PPO-specific parameters.
- Table 3.5: MCTS-specific parameters.
- Table 3.6: Shared SMC parameters.
- Table 3.7: Parameters for our extended agent.

We underline all default values in bold, all other parameter values indicated in the sets were run in an exhaustive grid for the ablations. The ablation results then report marginal performance over configurations and over seeds (repetitions). These experimental design decisions closely follow the suggestions laid out in the work by Patterson et al. (2024).

Despite conflating all model parameters into one joint set θ, we used separate neural network parameters for the policy, state, and state-action value models. Given the current dataset (replay buffer) within the training loop $\mathcal{D}_{(n)}$, the loss is a simple empirical cross-entropy of collective terms,

$$\mathcal{L}(\theta) = \mathbb{E}_{(S_t, A_t, \hat{q}_t, \hat{V}_t) \sim \mathcal{D}_{(n)}} \Bigg[\frac{c_v}{2} (\hat{V}_t - V_\theta^\pi(S_t))^2 + \frac{c_v}{2} (\hat{V}_t - Q_\theta^\pi(S_t, A_t))^2 - c_\pi \mathbb{E}_{a \sim \hat{q}_t} \ln \pi_\theta(a|S_t) - c_{ent} \mathcal{H}[\pi_\theta(a|S_t)] \Bigg] \tag{3.9}$$

where $\hat{q}_t$ and $\hat{V}_t$ are estimated targets for the policy and value respectively (see main text), and $\mathcal{H}[\pi_\theta] = -\mathbb{E}_{\pi_\theta} \ln \pi_\theta$ is an entropy penalty for the policy. We approximated $\mathcal{L}$ with stochastic gradient descent (see the hyperparameter tables), where we always used the AdamW optimizer (Loshchilov & Hutter, 2019) with an l_2 penalty of 10^{-6} and a learning rate of $3 \cdot 10^{-3}$. Gradients were clipped using two methods, in order: a max absolute value of 10 and a global norm limit of 10.

The replay buffer was implemented as a uniform circular buffer with its size calculated as:

$$\text{max-age of data} \times \text{number of parallel environments} \times \text{number of unroll steps}.$$

The max-age of data was tuned to fit into reasonable GPU memory.

For the A2C baseline, hyperparameters are detailed in Bonnet et al. (2024), and for the SAC baseline, refer to Freeman et al. (2021). Additionally, Table 3.2 provides version information for key software packages, this also directs towards default hyperparameters of baselines not listed here. We implemented everything based on Jax 0.4.30 in Python 3.12.4.

Table 3.2: Software module versioning that we used for our experiments (also includes default parameter settings).

Package	Version
brax	0.10.5
optax	0.2.3
flashbax	0.1.2
rlax	0.1.6
mctx	0.0.5
flax	0.8.4
jumanji	1.0.1

3.B.4 Neural Network Architectures

Neural network designs were largely adapted from the A2C reference (Bonnet et al., 2024) and Macfarlane et al. (2024). Minor modifications were made to handle the heterogeneity in environment action spaces. Notably, we standardized network architectures across environments wherever feasible, adjusting input embedding and output construction as needed. The specific configurations are listed below.

Brax Environments

- 2-layer MLP with 256 nodes per layer.
- Leaky-ReLU activations followed by LayerNorm.
- Outputs parameterized a diagonal multivariate Gaussian squashed via Tanh, as in Haarnoja et al. (2018).

Jumanji RubiksCube Environment

- 2-layer MLP with 256 nodes per layer (same as Brax).

- We used a flat representation for the 3-dimensional categorical action space (logits over all item-combinations). In contrast: the A2C baseline used a structured representation with three separate categorical outputs (logits per item).

Jumanji Snake Environment

- 2-layer MLP with 128 nodes per layer and Leaky-ReLU activations, followed by LayerNorm for the main module.
- Based on Bonnet et al. (2024), the input image is embedded using a single 3x3 convolutional layer with:
  - 3 channels.
  - Leaky-ReLU activation.
  - No LayerNorm.
- The resulting embedding was flattened before being passed to the main MLP module.

Table 3.3: Shared experiment hyperparameters.

Name	Symbol	Value Jumanji	Value Brax
SGD Minibatch size		256	256
SGD update steps		100	64
Unroll length (nr. steps in environment)		64	64
Batch-Size (nr. parallel environments)		128	64
(outer-loop) TD-Lambda	λ	0.95	0.9
(outer-loop) Discount	γ	0.997	0.99
Value Loss Scale	c_v	0.5	0.5
Policy Loss Scale	c_π	1.0	1.0
Entropy Loss Scale	c_{ent}	0.1	0.0003

Table 3.4: Proximal Policy Optimization hyperparameters. We did not use advantage-normalization computed the policy entropy exactly.

Name	Symbol	Value Jumanji	Value Brax
Policy-Ratio clipping	ϵ	0.3	0.3
Value Loss Scale	c_v	1.0	0.5
Policy Loss Scale	c_π	1.0	1.0
Entropy Loss Scale	c_{ent}	0.1	0.0003

Table 3.5: Gumbel Monte-Carlo tree search experiment hyperparameters (Danihelka et al., 2022).

Name	Symbol	Value Jumanji	Value Brax
Replay Buffer max-age		64	64
Nr. bootstrap atoms π	B	30	30
Search budget	N	$\{16, 64\}$	$\{16, 64\}$
Max depth		16	16
Max breadth		16	16

Table 3.6: Shared Sequential Monte-Carlo hyperparameters (ours and Macfarlane et al., 2024). Bold values indicate those used in the main results, with the remaining values in the set being explored in the ablations.

Name	Symbol	Value Jumanji	Value Brax
Replay Buffer max-age		64	64
Planner Depth	m	$\{\mathbf{4}, \mathbf{8}, 16\}$	$\{\mathbf{4}, \mathbf{8}\}$
Number of particles	K	$\{\mathbf{4}, \mathbf{8}, 16\}$	$\{\mathbf{4}, \mathbf{8}\}$
Resampling period	r	$\{1, \mathbf{3}\}$	$\{1, \mathbf{3}\}$
Target temperature (env. reward scale)	T	$\{1.0, \mathbf{0.1}\}$	$\{1.0, \mathbf{0.1}\}$
Nr. bootstrap atoms π	B	30	30

Table 3.7: Trust-Region Twisted Sequential Monte-Carlo hyperparameters (i.e., ours only). The underlined values recover the base SMC.

Name	Symbol	Value Jumanji	Value Brax
(inner-loop) Retrace(λ)	λ_{SMC}	0.95	0.9
(inner-loop) Discount	γ_{SMC}	0.997	0.99
(outer-loop) Value mixing $\hat{V}_t$	σ	0.5 (0.0)	$\{0.0, \mathbf{0.5}, 1.0\}$
Estimation $\hat{q}^*$		{Dirac, **Mess.-Pass.**}	{Dirac, **Mess.-Pass.**}
Revived resampling		{False, **True**}	{False, **True**}
Proposal (adaptive) Constraint	ϵ_α	{0.0, **0.1**, 0.3}	{0.0, **0.1**, 0.3}

3.B.5 Details on Constrained Proposals

We solve the constrained program in Eq. 3.7 in the SMC planner in Algorithm 1 for each individual state-particle $S_t^{(i)}$. To deal with general action-spaces (e.g., continuous), we sample B atoms from the prior policy π_θ for uniform bootstrapping. As stated in Theorem 3.1 and accompanying text, the Lagrangian of this program can be used to define a Boltzmann policy,

$$L(q, \beta^{-1}, \eta) = \mathbb{E}_q Q_\theta^\pi + (\epsilon_\alpha - \beta^{-1} KL(q \| \pi_\theta)) + (1 - \eta), \tag{3.10}$$

where taking the partial derivatives and setting them to zero gives $q^* \propto \pi_\theta \exp \beta Q_\theta^\pi$, which is analytically normalizable (see Grill et al. (2020) for comparison). Given this distribution q^*, we found that the following minimization problem was the most numerically stable in finding the optimal temperature parameter β^{-1},

$$\min_{\beta^{-1}} \| \epsilon_\alpha - \mathbb{E}_{q^*}[\ln q^*(a|S) - \ln \pi_\theta(a|S)] \|_2^2, \tag{3.11}$$

which we solve with a bisection search. In combination with bootstrapping from π_θ, the above $\ln \pi_\theta(a|S)$ is essentially an entropy constraint on q^*. As stated in Subsection 3.3.1, we set ϵ_α adaptively based on some greediness tolerance $\alpha \in [0, 1]$.

3.B.6 Details on Figure 3.1

We adapted the top-figure with the colored and grayed out trajectories from Figure 2 of Svensson et al. (2015) to show the discarded data in a naive forward-only sequential Monte-Carlo (SMC) planner. We generated the two bottom figures using our own SMC planner implementation. The KL-divergence was evaluated by: running SMC at timestep 0 on a dummy environment, extracting the sampled policy as by Eq. 3.6 or from Section 3.3.3, projecting the sampled logits back to their original action-space, and calculating the KL-divergence of the optimal stochastic policy to the canonical (non-bootstrapped) logits from SMC. Most importantly, this environment had discrete actions and zero reward everywhere, making the dynamics function like an absorbing state. For this reason, the optimal (stochastic) policy is uniformly random with a value V^π of zero everywhere. The bottom-left of Figure 3.1, thus, visualizes the consequence of recursive bootstrapping, which degenerates the policy in terms of KL-divergence from the optimal policy. Our method does not incur this in the bottom-right aside from the effect caused by the particle budget.

3.C Pseudocode

We give a simplified overview of our method in Algorithm 2, which is an extended version of the one from the main paper (Algorithm 1). The pseudocode documents our specific contributions in comparison to the base SMC planner.

Note that the implementation for the online tracking of ancestor values, following Moral et al. (2010) is very similar to the eligibility trace known in reinforcement learning (Seijen & Sutton, 2014, Sutton & Barto, 2018). Fundamentally, this can be considered as initializing

Algorithm 2 Bootstrapped Particle Filter for RL (Our TRT-SMC extending Alg. 1).

Require: K (number of particles), m (depth), r (resampling period), α (proposal greediness)

Initialize:

- Ancestor identifier $\{J_1^{(i)} = i\}_{i=1}^K$
- States $\{S_1^{(i)} \sim p(S_1)\}_{i=1}^K$
- Reference States $\{\widetilde{S}_1^{(i)} \leftarrow S_1^{(i)}\}_{i=1}^K$ // Revived resampling: track last non-terminal states
- Weights $\{\widetilde{w}_0^{(i)} = 1\}_{i=1}^K$
- Ancestor Log-Probabilities $\{\hat{Q}_0^{(i)} = 0\}_{i=1}^K$ // Policy Inference

for $t = 1$ to m **do**

// TRT-SMC: Create a set of Trust-Region twisted proposal distributions

$\{q_t^{(i)} \mid$ where $q_t^{(i)}$ solves Eq. 3.7 for a given $\alpha, S_t^{(i)}\}_{i=1}^K$

// Default SMC: Update particles

$\{A_t^{(i)} \sim q_t^{(i)}(A_t|S_t^{(i)})\}_{i=1}^K$

$\{S_{t+1}^{(i)} \sim p(S_{t+1}|S_t^{(i)}, A_t^{(i)})\}_{i=1}^K$

$\{\widetilde{w}_t^{(i)} = \widetilde{w}_{t-1}^{(i)} \frac{\pi(A_t^{(i)}|S_t^{(i)})}{q(A_t^{(i)}|S_t^{(i)})} e^{R_t^{(i)}} \frac{\mathbb{E}\exp V_\theta^\pi(S_{t+1}^{(i)})}{\exp V_\theta^\pi(S_t^{(i)})}\}_{i=1}^K$

// Revived resampling: Track the last non-terminal states

$\{\widetilde{S}_{t+1}^{(i)} \leftarrow \begin{cases} \widetilde{S}_t^{(i)}, & \text{If } S_{t+1}^{(i)} \text{ is terminal,} \\ S_{t+1}^{(i)}, & \text{Otherwise,} \end{cases}\}_{i=1}^K$

// Policy inference & Value Estimation: online accumulation of ancestor statistics

$\{\mathcal{J}^{(j)} \leftarrow \{i \in [1,K] \mid J_t^{(i)} = j\}\}_{j=1}^K$

$\hat{Q}_t^{(j)} = \hat{Q}_{t-1}^{(j)} + \begin{cases} 0, & \mathcal{J}_t^{(j)} = \emptyset, \\ \ln\left(\frac{1}{|\mathcal{J}_t^{(j)}|} \sum_{i \in \mathcal{J}_t^{(j)}} \frac{\widetilde{w}_t^{(i)}}{\widetilde{w}_{t-1}^{(i)}}\right), & \mathcal{J}_t^{(j)} \neq \emptyset, \end{cases}$

// Default SMC: Bootstrap (periodically) through resampling

if t mod $r == 0$ **then**

Normalized probability vector: $\overline{w}_t[i] = \frac{\widetilde{w}_t^{(i)}}{\sum_{j=1}^K \widetilde{w}_t^{(j)}}$

$\{J_t^{(i)} \sim \text{Categorical}(\overline{w}_t)\}_{i=1}^K$

// Revived resampling: resample particles by their reference states, then reset the references.

$\{(S_{t+1}^{(i)}, A_t^{(i)})\}_{i=1}^K \leftarrow \{(\widetilde{S}_{t+1}^{(J_t^{(i)})}, A_t^{(J_t^{(i)})})\}_{i=1}^K$

$\{\widetilde{S}_{t+1}^{(i)} \leftarrow S_{t+1}^{(i)}\}_{i=1}^K$

$\{\widetilde{w}_t^{(i)} = 1\}_{i=1}^K$

end if

end for

// Return the estimated values for performing policy inference

return $\{\hat{Q}_m^{(j)}\}_{i=1}^K$

the eligibilities of the ancestor particles to one, and continuously decaying these eligibilities while accumulating value updates (i.e., without updating the ancestor-eligibility).

3.D Supplementary Results

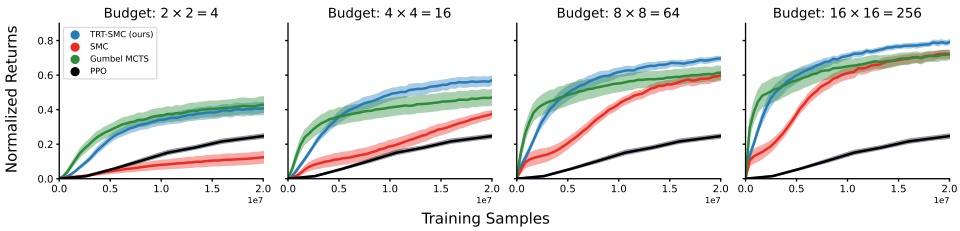


Figure 3.6: Normalized expected evaluation curves for the discrete environments over increasing planning budgets. These budgets were calculated as $N = K \times m$, where $K = m$, to keep the number of transitions uniform between the MCTS and SMC methods. Shaded regions give BCa $\alpha = 0.01$ intervals over 2 times 30 seeds. See also Figure 3.3 in the main text for a similar comparison to runtime scaling.

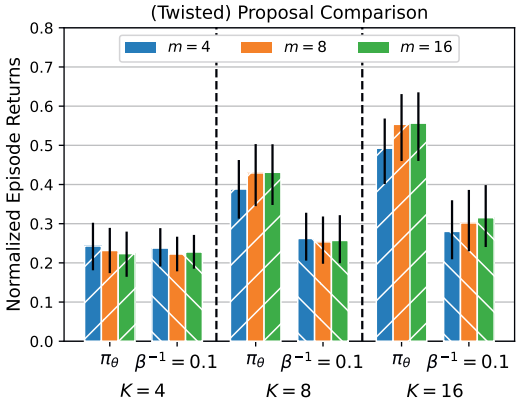


Figure 3.7: Final expected performance for the prior π_θ and the regularized proposal with a static temperature of $\beta^{-1} = 0.1$. This plot is identical to the left barplot in Figure 3.4 and shows that taking values into account for the proposal doesn't directly translate to improved performance (i.e., the approach taken by Lioutas et al., 2023). We found it essential for performance that the temperature β^{-1} must be set adequately, which we achieved with the adaptive trust-region method.

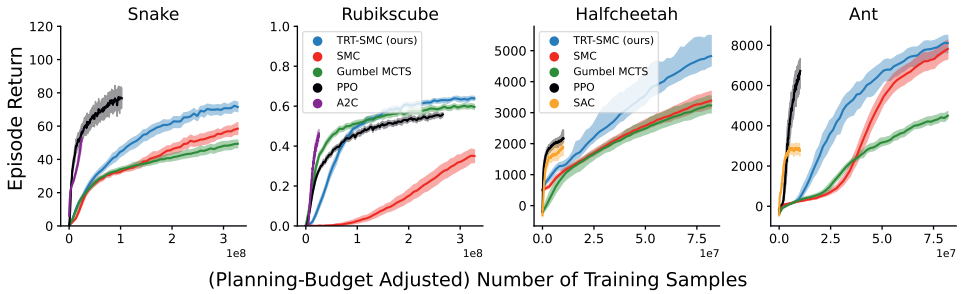


Figure 3.8: Main results from Figure 3.2 with the adjusted x-axis to account for additional true environment samples during planning. In terms of sample-efficiency this gives a stark contrast to the result of the main paper, and shows that the model-free methods are more sample efficient. However, methods that utilize planning can always utilize an accurate simulator to skew the x-axis similarly to the result of Figure 3.2.

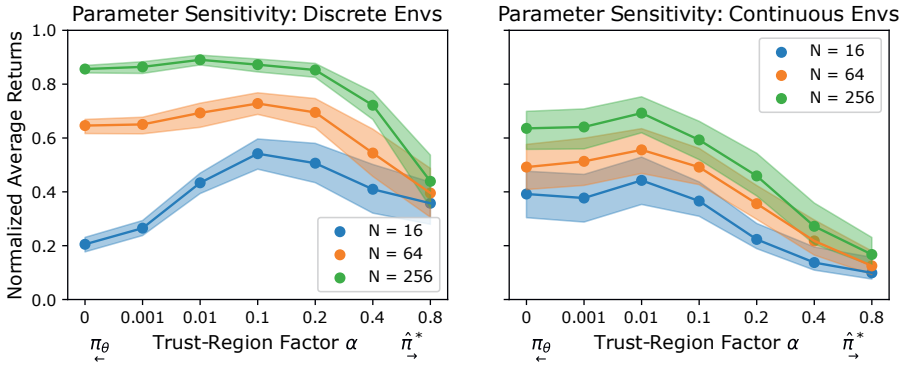


Figure 3.9: Sensitivity plot for the adaptive trust-region parameter α split for the discrete (Jumanji) and continuous (Brax) environments. This splits up the results shown in Figure 3.5 from the main paper.

4

Scalable Approximate Bayes-Adaptive Monte-Carlo Planning

There are some ideas so absurd that only an intellectual could believe them.

– Bertrand A. W. Russell, paraphrased from *My Philosophical Development* (1959)

A common theme in this manuscript has been that deep learning greatly helps in scaling intractable statistical computations. As stated in Chapter 1, this is a crucial ingredient for developing agents that can reliably perform sequential decision making under uncertainty. However, in Chapter 2 we found that end-to-end learning for statistical estimation typically does not converge to a robust agent. Furthermore, the expensive pretraining makes algorithm development a slow iterative process for practitioners. Thus, this chapter seeks better methods that 1) can train in faster runtime, and 2) converge to robust optimal agents.

Building on our insights of Chapter 3, this chapter develops a framework for approximating agents that optimally trade-off exploration and exploitation by combining sequential Monte-Carlo planning with meta-reinforcement learning. Based on Chapter 2 we derive a reliable loss function for learning a latent world model to steer planning under uncertainty. To specialize our method for GPU hardware, we tailor the loss to work efficiently on fixed-size data subsets.

Since the meta-RL setting requires us to learn on trajectory data, we find that the neural network learning induces an even greater bottleneck than in typical reinforcement learning settings. Thus, similarly to Chapter 3, we find that online planning considerably improves this bottleneck by shifting compute resources to the planner. As a consequence we are able to train optimally exploring agents with much faster runtime and with greater data efficiency.

The contents of this chapter were submitted to UAI 2026 with preprint (de Vries et al., 2026).

4.1 Introduction

Reinforcement Learning (**RL**) describes how to find an optimal policy for sequential decision making by maximizing expected returns under uncertainty (Sutton & Barto, 2018). In recent years, this framework has enjoyed many successes such as achieving superhuman performance in boardgames (Silver et al., 2018), and displaying rapid adaptation to test tasks (Bauer et al., 2023). The crux of this methodology is to optimally trade off gathering informative data (exploration) with focusing on data that guarantee high expected returns (exploitation). This is especially relevant in risk sensitive applications, since exploration and exploitation in an inaccurate model of the world can often incur significant repercussions.

A Bayes-optimal agent achieves an optimal balance of exploration and exploitation (Duff, 2002). It does so by augmenting its state-space with a belief over hypothetical environments given past data (meaning the current trajectory's history). It then selects its actions by accounting for the expected returns but also on how its own uncertainty evolves (the belief state). Unfortunately, computing this agent is hopelessly intractable due to the many nested integrals one has to deal with, this includes the marginalization over prior reward and transition functions at every new datapoint. Rightfully so, this has motivated development of heuristics like posterior sampling (Osband et al., 2013) or optimism (Munos, 2014). However, such heuristics typically sacrifice optimizing the true desired objective for the sake of tractability.

Instead, our focus is on direct approximations of the Bayes-optimal agent. Recently, variational approximations (Bishop, 2007) combined with amortized probabilistic inference (Margossian & Blei, 2024) using deep neural networks have shown great promise in scaling to high dimensional problem settings. For instance, this has been successfully tried for approximating beliefs (Becker et al., 2024, Igl et al., 2018, A. Mnih & Gregor, 2014), policy inference (Haarnoja et al., 2018, Piché et al., 2019), and Bayes-optimal agents (Iqbal et al., 2024, Zintgraf et al., 2021).

A key element in developing general **RL** algorithms, including approximate Bayes-optimal algorithms, that scale to higher dimensions, larger datasets, and greater (distributed) compute resources is careful engineering to match the intended hardware. This aspect is often less discussed but plays a subtle and essential role. This is especially relevant for approximating Bayes-optimal agents due to their history dependence. In deep learning, history dependence is typically tackled with recurrent neural network (RNN) architectures which require unrolling and BackPropagation Through Time (BPTT) for training with stochastic gradient descent. Unfortunately, this unrolling becomes expensive in memory for long sequences, is difficult to parallelize, and is also often difficult to train reliably (Goodfellow et al., 2016). Furthermore, they often require vast amounts of data and large model sizes in order to find a decent policy (Bauer et al., 2023). Our Bayes-optimal control problem fits within this description. Bayes optimal control requires a single agent to infer the true MDP, solve (possibly infinitely) many Markov decision processes (MDP; Puterman, 1994), and anticipate consequences of exploratory actions; leading to a much more difficult learning task compared to solving a single MDP. Thus, pushing the frontiers of current methodologies necessitates the development of frameworks that increase the scaling capabilities of throughput, whether it is in computation, data, or memory. Fortunately, recent frameworks like PureJaxRL (Lu et al., 2022) help in scaling up throughput of training by

dispatching large computations on GPUs which excel at highly parallelized data processing.

Transformer architectures (Vaswani et al., 2017) have recently been established as alternatives to RNNs, often showing much stronger performance while reducing training time due to improved parallelization over recurrent BPTT. Unfortunately, a key limitation of efficient GPU based frameworks is their strict adherence to computations on static shapes, whereas many history dependent control problems vary in input length. This is a strong restriction for utilizing more scalable and performant architectures such as Transformers (Parisotto & Salakhutdinov, 2021, Vaswani et al., 2017). Unlike RNNs, Transformers do not maintain an explicit state mechanism that can be cached throughout fixed size computations, they strictly require dynamically sized memories which cause overhead in GPU dispatch for resizing. Alternatively, they require padding of the history with empty observations, which also introduces unnecessary overhead. Interestingly, recent theoretical work (A. Gu & Dao, 2024, Katharopoulos et al., 2020) shows that Transformers can be viewed as a form of state-space models. This perspective suggests that linear state dynamics (of the form $x_t = Ax_{t-1} + Bu_t + C$) are the key ingredient behind scalable training on very long sequences, thanks to the associativity of linear recurrences. Furthermore, recent developments in state-space sequence models for RL (Becker et al., 2024, Lu et al., 2023, J. T. Smith et al., 2023) have drastically improved their performance compared to previously popular recurrent neural network (Duan et al., 2016) or Transformer based approaches (Parisotto & Salakhutdinov, 2021), at much improved training and inference computation time. For instance, state-space models can perform efficient inference in $\mathcal{O}(1)$ time, yet they are also trainable in $\mathcal{O}(\log n)$ time (n being the sequence length) by using GPU parallelization. Instead, typical recurrent network training scales with $\mathcal{O}(n)$ time and Transformer inference scales with $\mathcal{O}(n^2)$ (Vaswani et al., 2017) or $\mathcal{O}(n)$ with KV-caching (Pope et al., 2023).

Building on these considerations about computational efficiency, we next consider a complementary strategy with model-based RL. In MDP settings, a consensus is that model-based RL (Hafner et al., 2020, Sutton & Barto, 2018) improves data-efficiency (sample-efficiency) of algorithms by spending on average more compute per datapoint compared to model-free approaches. However, this dissertation has also shown that model-based RL utilizing online **sequential Monte-Carlo (SMC)** planning can improve *algorithm runtime* compared to model-free RL (Chapter 3) when a model of the environment is available. This is specifically relevant when environment interactions are also comparatively cheap to model training (gradient descent), since online planning often improves the quality of training data and learning targets (Hamrick et al., 2021). In typical MDPs this bottleneck depends strongly on the environment and the agent's model size, however, when specifically dealing with history dependent policies, this bottleneck can quickly shift in favor of taking more environment samples. The reason for this is that training on long trajectories at least requires a $O(\log n)$ time complexity for state-space models, whereas increasing the planning budget at decision time only amounts to a bigger constant in the $O(1)$ inference complexity[1]

Towards improving the scalability of training Bayes-adaptive agents, this paper in-

[1]Naturally, the planning budget (which induces the constant) should be adjusted based on the length of the trajectory. However, the optimal trade-off between planning budget, unroll length (or better yet, SGD budget) is a different research question we defer for future work.

troduces a new algorithm which we dub VariBASeD. VariBASeD combines a sequential Monte-Carlo planner in combination with amortized variational inference (Agrawal & Domke, 2021) to reliably train a Bayes adaptive agent. In essence, we phrase our Bayes adaptive agent as an intractable *target* distribution from which we draw approximate samples using more manageable *proposal* distributions that we design through efficient particle-filter methods (Chopin & Papaspiliopoulos, 2020). We show that online **SMC** planners can improve both the sample- and runtime-efficiency for training agents that approximate Bayes-optimal exploration and exploitation when compared to model-free methods like RL^2 (Duan et al., 2016). In summary our contributions are:

1. **Theoretical Framework:** We derive a history-dependent Expectation-Maximization (EM) method for approximate policy iteration with learned (amortized) belief states.
2. **Algorithmic Integration:** We develop a history-dependent importance-sampling strategy that integrates SMC-planning with the learned beliefs.
3. **Hardware-Efficient Implementation:** We design a specialized single-GPU training setup that handles episode boundaries via fixed-size batch sampling, enabling high-throughput GPU training with consumer-grade hardware.

To efficiently simulate downstream usage of Bayes-adaptive agents, we evaluated our contributions in a zero-shot multitask RL setting (Kirk et al., 2023). In this setting, we want the agent to maximize expected return on an unknown distribution over MDPs over *multiple* adaptation episodes (Duan et al., 2016). In other words, the agent's sequence model and value targets are accumulated for multiple "inner-episodes", they are only reset when the "true" underlying task is also reset. We evaluate our method on a continuous function optimization problem and a discrete gridworld environment with uncertain rewards, and observe improved sample-efficiency and runtime as we scale up the planner's compute budget.

4.2 Background

We want to find an optimal policy π for a sequential decision-making problem for which there is uncertainty on the reward or transition function. Our formalization relies on an infinite-horizon Markov decision process (MDP) (Puterman, 1994), which we write as a tuple $M = \langle \mathcal{S}, \mathcal{A}, p_R, p_S, \rho \rangle$. We have states $S \in \mathcal{S}$ and actions $A \in \mathcal{A}$, which generate rewards $p_R(R_t|S_t, A_t) \in \mathbb{P}(\mathbb{R})$ and transitions $p_S(S_{t+1}|S_t, A_t) \in \mathbb{P}(\mathcal{S})$ with initial state distribution $S_1 \sim \rho(S_1)$. Together, we write the realization for a finite sequence as $h_{1:T} = \{a_t, r_t, s_{t+1}\}_{t=1}^{T}$ and assume that $h_{1:\infty}$ always ends up in an absorbing state with zero rewards (this is a weak assumption that we satisfy by discounting the returns that act as a termination probability $1 - \gamma \in [0, 1]$). Our aim is to find a policy π that satisfies, $\max_\pi \mathcal{J}_M(\pi) = \max_\pi \mathbb{E}_{\pi, p_R, p_S, \rho}[\sum_{t=1}^{\infty} R_t]$.

Bayesian Reinforcement Learning. In our setting we assume uncertainty on the true MDP $M_{true} \sim p(M)$, particularly we are uncertain about the true reward and state dynamics p_R, p_S. This creates a non-Markovian RL problem due to induced history dependence for

inferring M. Bayesian RL methods (Duff, 2002, Ghavamzadeh et al., 2015) offer a useful framework to deal with history dependence by maintaining a posterior of MDPs over time $p(M_t|H_{<t})$. To include this, write the joint density for a finite sequence as

$$p_\pi(H_{1:T}, M_{1:T}) = \mathbb{E}_{\rho(S_1,M_0)} \prod_{t=1}^{T} p(S_{t+1}, R_t|A_t, S_t, M_t) \cdot \pi(A_t|S_t, M_t) \cdot p(M_t|H_{<t}), \tag{4.1}$$

where $\rho(S_1, M_0) = \rho(S_1|M_0)p(M_0)$ factorizes as some conditional initial state density with a known MDP-prior. Note that we combined S_{t+1}, R_t for brevity, they are in principle independent.

In essence, the above density imposes structure to the marginal distribution for $p_\pi(H_{1:T})$ by augmenting the dynamics with an evolving posterior distribution over MDPs. For this reason, the posterior is often summarized in terms of a belief state $b_t = p(M_t|H_{<t})$ that augments the dynamics of the base MDP. We can write this new MDP as $M^+ = \langle \mathcal{S}^+, \mathcal{A}, p_R^+, p_S^+, \rho^+ \rangle$, where $\mathcal{S}^+ = \mathcal{S} \times \mathcal{B}$ is the belief $b_t \in \mathcal{B}$ augmented state-space, such that the dynamics, rewards, and policy satisfy,

$$p_S^+(b_{t+1}, S_{t+1}|S_t, A_t, b_t) = \delta(b_{t+1} = p(M_{t+1}|H_{<t} \cup H_t))\mathbb{E}_{M_t \sim b_t} p(S_{t+1}|A_t, S_t, M_t),$$

$$p_R^+(R_t|S_t, A_t, b_t) = \mathbb{E}_{M_t \sim b_t} p(R_t|A_t, S_t, M_t), \quad \pi^+(A_t|S_t, b_t) = \mathbb{E}_{M_t \sim b_t} \pi(A_t|S_t, M_t).$$

The augmented MDP M^+ is also known as a Bayes-adaptive MDP (BA-MDP), for which the optimal policy $\pi^+ : \mathcal{S} \times \mathcal{B} \to \mathbb{P}(\mathcal{A})$ is a *Bayes-optimal policy* (Duff, 2002). In other words, a policy that optimally trades-off exploration and exploitation. This differs from common heuristics like posterior sampling or optimism (Osband & Roy, 2017, Osband et al., 2013) since π^+ explicitly accounts for how the complete belief evolves given additional data and its impact on expected returns.

Belief Amortization through Meta-Learning. Finding π in the augmented state-space $\mathcal{S} \times \mathcal{B}$ is, unfortunately, intractable for most interesting problems. Furthermore, tracking the belief state b_t itself is usually also intractable, assumes knowledge of the density for the reward and state dynamics, and requires specifying a reasonable prior b_0. To deal with this intractability, a successful approach is to use amortization (Amos, 2023) in combination with flexible function approximation. For this, we assume that we can approximate the true belief, reward, and state dynamics, and the policy, with e.g., a large neural network $h_\theta(\cdot)$. Then we perform inference to the probabilistic graphical model of Eq. (4.1) by fitting the neural network parameters θ directly on (a proxy of) the evidence $p_\pi(H_{1:T})$. In other words, amortization means that we can rely on comparatively cheap neural network evaluations in downstream tasks that approximate probabilistic inference to our (intractable) model after performing an expensive training procedure.

A flexible setup for such a training procedure is through meta learning (Mikulik et al., 2020), which only requires us to be able to sample environments from a distinct generator of MDPs $\mathcal{G}_{train}(M)$, not to be confused with the MDP prior $p(M)$. For brevity, let us write the joint density of Eq. (4.1) as $p_\pi(H_{1:t}, M_{1:t}|\theta)$ so that each component now depends on neural network parameters θ. We will denote the belief b_t as a parametric

family $b_{\phi_t} = p(M|\phi_t = h_\theta(H_{<t}))$, where our neural network h_θ maps histories to the belief parameters ϕ_t. The idea is to compute the policy objective $\hat{\mathcal{J}}_M(\theta) = \mathbb{E}_{p_\pi(H_{1:\infty}|\theta)}[\sum_{t=1}^{\infty} R_t]$ in expectation of the environment generator, and perform stochastic gradient ascent on the neural network parameters for h_θ. This assumes end-to-end differentiability for θ, where we can write the optimization objective as the l.h.s. of the lower-bound,

$$\max_\theta \mathbb{E}_{M\sim\mathcal{G}_{train}(M)}[\hat{\mathcal{J}}_M(\theta)] \leq \mathbb{E}_{M\sim\mathcal{G}_{train}(M)}[\max_\pi \mathcal{J}_M(\pi)], \tag{4.2}$$

which can, e.g., be optimized through Monte-Carlo combined with REINFORCE (R. J. Williams, 1992) and backpropagation through time (Duan et al., 2016). Effectively, fitting θ in the l.h.s. of Eq. (4.2) distills the uncertainty induced by the true rewards and transitions of sample MDPs $M \sim \mathcal{G}_{train}$ into the predicted belief parameters $\phi_t, \forall t$. Thus, we obtain a Bayes-optimal policy (at θ^*) for a prior that has a learned inductive bias (Finn et al., 2017) to the MDPs following $\mathcal{G}_{train}$.

When we directly model the density $p_\pi(H_{1:t}, M_{1:t}|\theta)$, we make the assumption that the posterior over MDPs $b_{\phi_t} = p_\pi(M_t|h_\theta(H_{<t}))$ can be captured by our model. Even when our neural networks are sufficiently expressive, it often helps to add structure to the modeling problem. In this regard, variational approximations (Igl et al., 2018, Kingma & Welling, 2014, A. Mnih & Gregor, 2014, Zintgraf et al., 2021) can greatly help as they only assume that we can model a simplified posterior $b_t^q \approx b_t$, or with neural networks $b_{\phi_t}^q = q_\theta(M_t|H_{<t}) \approx p(M_t|H_{<t})$. For variational distributions q_θ it is easier to assume end-to-end differentiability due to simplifying assumptions on the parametric family, e.g., enabling reparameterized gradients (Kingma & Welling, 2014). The variational parameters for $b_{\phi_t}^q$ are then fitted by augmenting the objective $\mathcal{J}_M$ with a regularized decoding loss. Our method also adopts this variational Bayes framework for the MDP posterior, which we detail in Section 4.3.2.

Finally, an important distinction between meta-RL and conventional RL is that the augmented dynamics for termination and episodic learning are handled differently (Duan et al., 2016, Zintgraf et al., 2021). In particular, if a sequence of states S_t ends up in an absorbing state, which typically signals that the environment can be reset to a new initial state (i.e., a new episode), then this should not reset the belief state. The belief states are only reset when the true MDP changes. This means that episodic learning becomes an essential part of the augmented dynamics: both the returns from future episodes and data from previous episodes are taken into account when optimizing Eq. (4.2).

Expectation-Maximization. We will now consider optimizing the l.h.s. of Eq. (4.2) using Expectation-Maximization (EM) (Neal & Hinton, 1998) methods by leveraging the Control-as-Inference framework (Levine, 2018, Rawlik et al., 2012, Toussaint et al., 2008, Ziebart, 2010). These methods embed the control aspect as part of the probabilistic graphical model, which gives rise to a flexible energy-based formulation of the RL problem. This is in contrast to typical approaches, which treat the policy as separate from the environment (Puterman, 1994). Most importantly, this framework naturally enables regularized approximate policy iteration (Geist et al., 2019), which has proven extremely effective (Abdolmaleki et al., 2018, Macfarlane et al., 2024) compared to unregularized methods such as plain REINFORCE.

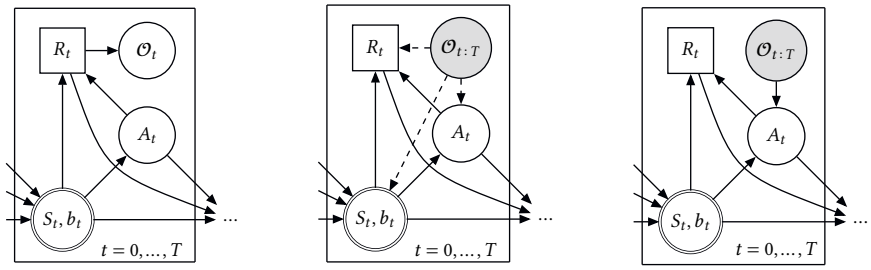


Figure 4.1: Generative model (left), and the unstructured (middle) vs. structured inference model (right) that we consider. Stochastic variables are placed in circles, deterministic variables in rectangles. The double circle for $\langle S_t, b_t \rangle$ indicates a "joint" variable, however, note that S_t is stochastic whereas the belief evolves deterministically. Colored nodes are observed, blank nodes are latent.

4

In summary, an EM-style optimization framework can be formulated in our setup by firstly augmenting the generative density $p_\pi(H_{1:T}, M_{1:T})$ (without parameters θ for brevity, and using the true belief b_t) with a binary random variable $\mathcal{O} \in \{0, 1\}$ for each t. We will assume $\mathcal{O}_t \perp\!\!\!\perp H_{1:T} | R_t$, with a choice of likelihood $p(\mathcal{O}_t = 1 | S_t, b_t, A_t) \propto \mathbb{E}_{p(R_t|S_t,b_t,A_t)} \exp R_t$. Note however, that we will now assume deterministic rewards, $p(R_t | S_t, b_t, A_t) = \delta(R_t - f_R(S_t, b_t, A_t))$, we provide a brief discussion on reward stochasticity in Appendix 4.B.1. For a finite sequence of data, this corresponds to the generative model in the left of Figure 4.1. When we model the inverse problem by conditioning on $\mathcal{O}_{1:T} = 1$, naively we obtain an exponentially tilted density,

$$p(H_{1:T}, M_{1:T} | \mathcal{O}_{1:T} = 1) \propto p(\mathcal{O}_{1:T} | H_{1:T}, M_{1:T}) p(H_{1:T}, M_{1:T}) = \Phi_{1:T} p(H_{1:T}, M_{1:T}), \quad (4.3)$$

summarizing the likelihood for $\mathcal{O}_{1:T}$ in terms of a potential $\Phi_{1:T}$. To be clear, we abbreviate notation for $\mathcal{O}_t = 1$ simply as $\mathcal{O}_t$, or $\mathcal{O}_{a:b} = 1$ as $\mathcal{O}_{a:b}$ for sequences. This inference problem is illustrated in the middle of Figure 4.1. Estimating the parameters θ for this naive model will lead to optimistic learning of the transitions, rewards, and belief (Levine, 2018) as the true densities of these components will be shifted towards the potential. Instead, we consider structuring this inference problem, by focusing on the tilted policy π^+.

Theorem 4.1 (Proof App. 4.A.1)**.** *The policy $p^+_\pi(A_t | S_t, b_t, \mathcal{O}_{t:T}) \propto \Phi_{t:T} \pi^+(A_t | S_t, b_t)$ satisfies,*

$$q^* = \arg\max_q \mathcal{J}_{proxy}(q, \pi^+) = \arg\max_q \mathbb{E}_{p_q} \left[\sum_{t=1}^{T} R_t - KL(q(A_t | S_t, b_t) \| \pi^+(A_t | S_t, b_t)) \right] \quad (4.4)$$

which is a regularized policy objective, where q^* also guarantees a policy improvement in the unregularized MDP, $\mathbb{E}_{p_{q^*}}[\sum_{t=1}^T R_t] \geq \mathbb{E}_{p_{\pi^+}}[\sum_{t=1}^T R_t]$.

The inference problem posed by Theorem 4.1 belongs to the graphical model on the right of Figure 4.1, and gives us a proxy objective $\mathcal{J}_{proxy}(q, \pi^+)$ for the RL-problem that regularizes q^* to π^+ in KL-divergence. We can use this to naturally formulate an approximate policy iteration algorithm through the EM-framework, as given in Proposition 4.2.

Proposition 4.2 (Proof App. 4.A.2)**.** *Starting from some prior $\pi^+_{(1)}$ with sufficient density everywhere, at each iteration i, alternate the following: E-step: solve $\max_q \mathcal{J}_{proxy}(q, \pi^+_{(i)})$ to get*

$q^*_{(i+1)}$ by fixing the current prior $\pi^+_{(i)}$; M-step: distill the solution $q^*_{(i+1)}$ into a new prior $\pi^+_{(i+1)}$ by solving $\max_{\pi^+} \mathcal{J}_{proxy}(q^*_{(i+1)}, \pi^+)$. *The sequence $\{\pi^+_{(i)}\}_{i=1}^N$ converges to the Bayes-optimal policy.*

Actually solving each alternating step for the EM-procedure of Proposition 4.2 is usually intractable. To mitigate this, the M-step can be approximated by parameterizing the prior policy with parameters π^+_θ and performing a few steps of gradient ascent on $\nabla_\theta \mathcal{J}_{proxy}(q^*_{(i+1)}, \pi^+_\theta)$ (Bishop, 2007). In MDP-algorithms like MPO (Abdolmaleki et al., 2018), the E-step is approximated by estimating the objective $\mathcal{J}(q, \pi)$ through learning of a value function $Q^{q,\pi}_{soft}(S_t, A_t)$ from sample rollouts. The 'soft' subscript emphasizes that this value function estimates the q to π KL-regularized value. Then, a 1-step optimization problem is performed to obtain sample solutions to the tilted policy $\hat{q}^* \approx \pi \exp Q^{q,\pi}_{soft}$ analogous to Theorem 4.1 over some stationary distribution of states (e.g., the replay buffer).

4.3 Bayes-Adaptive Particle-Filter Planning

We present an Expectation-Maximization (EM) framework to learn solutions to Bayes-adaptive MDPs through a combination of variational Bayes, sequential Monte-Carlo planning, and meta-learning. On a base level, we make a considerable improvement to the E-step by addressing a bottleneck induced by gradient-based learning of variational belief-states. Instead of directly sampling from our policy as done in model-free methods, which is cheap, we shift more of the computation to online planning to generate higher quality data and learning targets. This improves scaling to compute resources for learning adaptive agents compared to model-free approaches, because we assume that the bottleneck of training is induced by stochastic gradient descent for which planning can enhance this procedure at reasonable cost. This is a reasonable assumptions when training with long histories due to the $O(1)$ inference cost of state-space sequence models at decision-events, but at least $O(\log n)$ costs during BPTT. To achieve this, we adapt the Control-as-Inference framework (Levine, 2018, Ziebart, 2010) to our setting to derive a new sequential importance resampling procedure (Piché et al., 2019) in Section 4.3.1 and an EM-formulation for belief learning in Section 4.3.2.

4.3.1 Improved E-step with Belief-SMC planning

In MDP-settings, typical methods (Abdolmaleki et al., 2018, Macfarlane et al., 2024) make strong use of the duality from Theorem 4.1 to obtain approximate solutions, it implies that the optimal regularized policy requires us to infer the tilted policy $p^+_\pi(A_t|S_t, b_t, \mathcal{O}_{t:T}), \forall s \in \mathcal{S}$. We will also use this duality to sample solutions $\hat{q}^*_t \approx q^*_t = p^+_\pi(A_t|S_t, b_t, \mathcal{O}_{t:T})$ through a particle-filter planner with variational beliefs, which we can subsequently use as learning targets in the M-step. As stated in the Introduction, we want to perform importance-sampling to an intractable *target* distribution with a more manageable *proposal* distribution, and particle-filter methods provide tools on how to efficiently achieve this (Chopin & Papaspiliopoulos, 2020).

Suppose we are interested in the cumulative returns of an agent over a window, $f(H_{1:t}) = \sum_{i=1}^t R_i$, under the target distribution given in Eq. (4.3). We can then write

importance-sampling weights as,

$$\mathbb{E}_{\Phi_{1:T}p_\pi} f(H_{1:t}) = \mathbb{E}_{p_{q,b^q}(H_{1:T},M_{1:T})} \frac{p_\pi(H_{1:T},M_{1:T}|\mathcal{O}_{1:T})}{p_{q,b^q}(H_{1:T},M_{1:T})} f(H_{1:t}) = \mathbb{E}_{p_{q,b^q}}[w_t f(H_{1:t})] \quad (4.5)$$

where the proposal $p_{q,b^q}(H_{1:T},M_{1:T})$ is similarly defined as Eq. (4.1),

$$p_{q,b^q}(H_{1:T},M_{1:T}) = \mathbb{E}_{\rho(S_1,M_0)} \prod_{t=1}^{T} p(S_{t+1},R_t|A_t,S_t,M_t)\cdot q(A_t|S_t,M_t)\cdot q(M_t|H_{<t}), \quad (4.6)$$

where we assume knowledge of $\rho(S_1,M_0)$. We call $b_t^q(M) = q(M_t|H_{<t})$ the variational belief with b_t the true belief, and $q(A_t|S_t,M_t)$ the variational policy with π^+ the prior policy. The idea is that we sample sequences according to our variational model q, and weight the resulting statistics according to their importance weights w_t.

Corollary 4.3 (Proof Appendix 4.A.3)**.** *The importance sampling weights w_t for drawing approximate samples from Eq. (4.3) using the proposal distribution from Eq. (4.6) can be expressed as,*

$$\frac{w_t}{w_{t-1}} = \mathbb{E}_{b_t^q}\mathbb{E}_{p_q}\left[\frac{\mathbb{E}_{b_t^q}\omega_t p_\pi(H_t|S_t,M_t)}{\mathbb{E}_{b_t^q} p_q(H_t|S_t,M_t)}\cdot \exp R_t \cdot \frac{\mathbb{E}_{b_{t+1}^q}\omega_{t+1}\exp V_{soft}^{q,\pi}(S_{t+1},M_{t+1})}{\mathbb{E}_{b_t^q}\omega_t \exp V_{soft}^{q,\pi}(S_t,M_t)}\cdot f(H_{1:t})\right]$$

where $\omega_t = \frac{b_t(M)}{b_t^q(M)}$ are nested importance-sampling weights to adjust for the variational beliefs.

There are two main limitations to evaluating the weights in Corollary 4.3, we need the regularized value functions $V_{soft}^{q,\pi}$ and the true belief b_t to evaluate the nested weights ω_t. To deal with this, we will learn the value functions V_θ using Temporal-Difference methods (Lawson et al., 2018, Sutton & Barto, 2018) as is typical in SMC-planners (de Vries, He, Oren, & Spaan, 2025, Lioutas et al., 2023, Macfarlane et al., 2024, Piché et al., 2019). Then, we will approximate the nested weights $\hat{\omega}_t \approx \omega_t$ with a single-timestep correction derived from the belief state recursion,

$$b_{t+1}(M) \propto b_t(M)\cdot p(H_t|M) \approx b_t^q(M)\cdot p(H_t|M) \quad (4.7)$$

In other words, we will treat the previous variational belief b_t^q as if it were the true belief at the previous timestep. This is also a common heuristic employed in variational sequence models to regularize their model's predictions in a way that remains computationally feasible (Hafner et al., 2020). From the belief recursion, we reuse importance sampling tools to write,

$$\mathbb{E}_{b_t^q}\frac{b_t(M_t)}{b_t^q(M_t)}\exp V_{soft}^{\pi}(S_t,M_t) \approx \sum_{j=1}^{K_\Omega} \bar{\omega}_t^{(j)} \exp V_{soft}^{\pi}(S_t,M_t), \quad M_t \sim b_t^q, \quad (4.8)$$

$$\bar{\omega}_t^{(j)} = \text{softmax}(\hat{\omega}_t)^{(j)}$$

$$\ln \hat{\omega}_t^{(j)} = \ln \hat{b}_{t-1}^q(M_t^{(j)}) + \ln p(H_t|S_t,M_t^{(j)}) - \ln b_t^q(M_t^{(j)})$$

Algorithm 3 Inner-Loop; BA-SMC planner

Require: K, K_Ω (nr. of particles), m (depth)
Require: S (initial state), ϕ (initial belief)
Initialize hyperstates $\{S_1^{(i)}, \phi_1^{(i)} = S, \phi\}_{i=1}^K$
Initialize weights $\{\widetilde{w}_0^{(i)} = \bar{\omega}_1^{(i)} = \frac{1}{K}\}_{i=1}^K$
for $t = 1$ to m **do**
  // Update particles (history)
  $\{A_t^{(i)} \sim q(A_t|S_t^{(i)}, \phi_t^{(i)})\}_{i=1}^K$
  $\{R_t^{(i)} \sim p(R_t|S_t^{(i)}, \phi_t^{(i)}, A_t^{(i)})\}_{i=1}^K$
  $\{S_{t+1}^{(i)} \sim p(S_{t+1}|S_t^{(i)}, \phi_t^{(i)}, A_t^{(i)})\}_{i=1}^K$
  $\{\phi_{t+1}^{(i)} = h_\theta(\phi_t^{(i)}, H_t^{(i)})\}_{i=1}^K$
  // Update nested-particles (belief)
  $\{M_t^{(i,j)} \sim b_t^q(M|\phi_{t+1}^{(i)})\}_{j=1}^{K_\Omega}, \forall i \in [1, K]$
  // Update importance sampling weights
  Compute $\{\bar{\omega}_{t+1}^{(i)}\}_{i=1}^K$ with Eq. (4.7)
  Compute $\bar{w}_t$ with Corollary 4.3
  // Bootstrap through resampling
  $\{(\phi_{t+1}^{(i)}, H_t(i))\}_{i=1}^K \sim \text{Mult}(K, \overline{w}_t)$
  $\{\widetilde{w}_t^{(i)} = 1\}_{i=1}^K$
end for
return $\{\phi_{1:m+1}^{(i)}, H_{1:m}^{(i)}, \widetilde{w}_{0:m}^{(i)}\}_{i=1}^K$

Algorithm 4 Outer-Loop; EM-framework

Require: Initial parameters $\theta_{(1)}$ and buffer $\mathcal{D}_{(1)}$
Require: MDP generator $\mathcal{G}_{train}$ & period p_M
for $n = 1$ to N **do**
  if $(n+1) \mod p_M = 0$ **then**
    Sample $M_{(n)} \sim \mathcal{G}_{train}$
    Reset belief representation $\phi_0^{(n)}$
  end if
  $S_1 \sim p_{M_{(n)}}(S_1), A_0 = R_0 = \emptyset$
  for $t = 1$ to T **do**
    Update $\phi_t^{(n)} = h_{\theta_{(n)}}(\phi_{t-1}^{(n)}, H_{t-1})$
    Infer policy $\hat{q}_t^* \sim \text{SMC}(S_t, \phi_t^{(n)})$
    Compute value $\hat{V}_t' \approx V_{\theta_{(n)}}^{\pi}(S_t, \phi_t^{(n)})$
    Sample action $A_t \sim \hat{q}_t^*$
    $S_{t+1}, R_t \sim p_{M_{(n)}}(S_{t+1}, R_t|S_t, A_t)$
    Add $(S_t, A_t, R_t, \phi_{t-1}^{(n)}, \hat{q}_t^*, \hat{V}_t')$ to $\mathcal{D}_{(n)}$
  end for
  Compute TD-estimators from $\mathcal{D}_{(n)}$
  Update $\theta_{(n+1)}$ with SGD
  (Optionally: Wrap $\mathcal{D}_{(n+1)}$ from $\mathcal{D}_{(n)}$)
end for
return $h_{\theta_{(N+1)}}, \pi_{\theta_{(N+1)}}^+$

where $M_t^{(j)} \sim b_t^q$, and $\ln \hat{\omega}_t^{(j)}$ are unnormalized belief-particles. Finally, we correlate (re-use) samples $M \sim b_t^q$ for each term involving ω_t in Corollary 4.3 to reduce variance between independent model components.

Our approach differs from POMDP methods in that we regenerate all nested particles $\omega_t^{(j)}$ at each timestep from the variational beliefs, rather than let them evolve them through a transition function (Abdulsamad et al., 2025). A full overview of our variational Bayes-adaptive planner is given in pseudocode in Algorithm 3. The pseudocode makes use of variational paramerization of the beliefs $b_{\phi_t}^q(M)$, with an inference function h_θ to generate ϕ_t, this part and the M-step (learning) are explained in the next section.

4.3.2 Gradient Based (EM)-step for Amortized Variational Bayes

The previous section described how we can sample approximate solutions $\hat{q}^*$ to the E-step for the policy π_θ. The EM-loop is then completed for π_θ by minimizing the empirical cross-entropy $\min_\theta \mathbb{E}[CE(\hat{q}^*, \pi_\theta)]$. However, our approach also relies on variational belief states b_t^q, which induces a joint optimization problem (i.e., for b^q and π_θ). Firstly, rather than estimating b^q at each timestep explicitly, we assume a simple parametric form with parameters ϕ_t, such that $b_t^q \leftarrow b_{\phi_t}^q$ where ϕ_t can be generated using an autoregressive inference function $\phi_t = h_\theta(\phi_{t-1}, H_{t-1})$. For instance, when $b_{\phi_t}^q$ is assumed to be Gaussian, we can use a recurrent neural network (Hochreiter & Schmidhuber, 1997) or state-space

model (Lu et al., 2023) for h_θ to predict the mean and covariance $\phi_t = (\mu_t, \Sigma_t)$ of this latent variational distribution.

Although approximating the E-step through sampling makes sense for π_θ due to the non-differentiable operations in the particle-filter planner (e.g., resampling in Algorithm 3), this is not necessarily the best strategy for the beliefs $b^q_{\phi_t}$ (S. Gu et al., 2015). To reduce variance, we will instead opt for an amortized gradient-based optimization of the E-step for obtaining ϕ^*_t (Kingma & Welling, 2014). This requires us to formulate a training objective through the evidence lower-bound in order to optimize for $\theta^*_{(n)}$, which conveniently reduces the M-step to a simple assignment $\theta_{(n+1)} \leftarrow \theta^*_{(n)}$. To formulate this E-step, we first state the following proposition, which is essentially an extension and rearrangement of Theorem 4.1.

Proposition 4.4 (Proof Appendix 4.A.4)**.** *The optimization objective* $\max_{b^q_t, q} \mathcal{J}((b^q_t, q_t), (b_t, \pi^+_t))$ *with now the variational and true belief respectively, can be written as,*

$$\mathbb{E}_{\rho_{q^\star}(H_{<t})} \max_{b^q_t} \mathbb{E}_{b^q_t} \left(\max_{q_t} \mathbb{E}_{q_t} \left[Q^{q^\star}(S_t, b_t, A_t) - KL(q_t \| \pi^+_t) \right] + \mathcal{L}_{\mathcal{Z}}(b^q_t | b_0) \right), \tag{4.9}$$

where $\mathcal{L}_{\mathcal{Z}}(b^q_t | b_0)$ *is another evidence lower-bound for the marginal* $p(H_{<t})$*, defined as,*

$$\begin{aligned} \mathcal{L}_{\mathcal{Z}}(b^q_{t+1} | b_0) = -KL(b^q_{t+1} \| b_0) + \mathbb{E}_{b^q_{t+1}} \Bigg[& \ln p_\theta(S_1 | M) \\ & + \sum_{i=1}^{t} \ln p_\theta(S_{t+1} | S_t, A_t, M) + \ln p_\theta(R_t | S_t, A_t, M) + \ln \pi_\theta(A_t | S_t, M) \Bigg], \end{aligned} \tag{4.10}$$

where we assume $b_0 = p(M_0)$ *to be a simple fixed prior (e.g., standard Gaussian).*

The above result is an important property as it shows the relation of policy search q_t to the variational belief search b^q_t. It shows that we can sample prior trajectories $H_{<t} \sim \rho_{q^*}$ from the optimal history-occupancy to then a) update our variational belief b^q_t and perform planning with to infer q^*_t (Algorithm 3), and b) perform gradient learning on $\mathcal{L}_Z$ to amortize the maximization.

From Prop. 4.4, we can motivate the following loss function to fit our joint neural net parameters θ to the graphical model depicted on the right of Figure 4.1,

$$\begin{aligned} \mathcal{L}(\theta, \mathcal{D}_{(n)}) = \mathbb{E}_{(\hat{V}_t, \hat{q}^*_t, H_{t-h:t}) \sim \mathcal{D}_{(n)}} \Big[& \frac{c_v}{2} (\hat{V}_t - V^\pi_\theta(S_t, \mathrm{sg}[\phi_t]))^2 - \\ & c_\pi \mathbb{E}_{A_t \sim \hat{q}^*_t} [\ln \pi^+_\theta(A_t | S_t, \mathrm{sg}[\phi_t]))] - c_b \mathcal{L}_{\mathcal{Z}}(\phi_t | \phi'_{t-h})] \end{aligned} \tag{4.11}$$

where $\phi_t = h_\theta(\phi_{t-h}; H_{t-h:t})$ is the unrolled variational parameters over a burn-in window, ϕ'_{t-h} is a cached hidden state, and $\mathbf{sg}[\cdot]$ is a stop-gradient operator for the reparametrized predictive for the value and policy networks. In other words, we have replaced the base prior b_0 from Prop. 4.4 with an intermediate prior 'h' steps before the current timestep t. This last unrolling and hidden state caching is important as it efficiently deals with varying length trajectories when optimizing this loss on the GPU, which requires fixed-size batches. However, the recurrent state caching can unfortunately drift in representation from the optimization process as noticed by Kirkpatrick et al. (2017). In practice, this typically

requires having longer unroll windows to deal with this mismatch, which necessitates tuning to trade off performance with runtime.

Our final heuristic we use in optimizing our loss function is to cancel the gradients for π_θ in the belief ELBO $\mathcal{L}_\mathcal{Z}(\phi_t|\phi'_{t-h})$. This means that the variational belief is regularized to the term $\ln \pi_{\text{sg}[\theta]}(A_t|S_t, M)$ in a way that does not influence the policy network (see Prop. 4.4). This is motivated by the fact that we are already learning π_θ on the sample policies from the SMC planner; we want to prevent a double learning effect while also regularizing the belief to support the policy predictions.

4.4 Experiments

We present results comparing our method, VariBASeD, against the model-free RL^2 baseline (Duan et al., 2016) on two environments: function optimization and gridworld exploration. We consider a multi-task setting with a uniform, but unknown, prior distribution over tasks. The specific details of this task distribution are described in Appendix 4.C.1 along with hardware requirements in Appendix 4.C.2, and algorithm hyperparameters in Appendix 4.C.3. Our experiments aim to demonstrate two key points: a) the ability of VariBASeD to achieve effective (in-domain) test-time adaptation, and b) that our method's performance scales with increased planning budget relative to both data and runtime complexity.

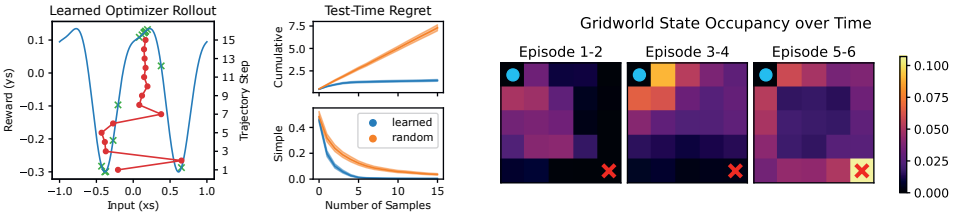


Figure 4.2: Evaluation rollouts on the function optimization and gridworld problems over multiple test-episodes. Due to the stochastic nature of our method, we visualize one sample rollout for the function optimization problem (with average metrics), and the averaged rollouts for the gridworld. For the gridworld, the circle indicates the starting-tile and the cross the goal-tile. The function optimization agent used a planning budget of $H = 1, K = 16$ and the gridworld agent used a budget of $H = 4, K = 32$.

Example Rollouts. In Figure 4.2, we first show example rollouts of a single fully trained agent on the two considered problems. Due to the stochastic nature of our particle planner, we provide a visualization of test rollouts for both a single sample rollout and the full distribution's state-occupancy. Specifically, for the function optimization problem, we show a single agent rollout (red line) whereas for the gridworld problem we show its rollout distribution in terms of the state-occupancy. Compared to a random baseline, our method demonstrates adaptive behavior on the test task in both environments. This is shown by the state-occupancy on the gridworld concentrating on the goal tile in episodes 5–6, and the sublinear cumulative regret achieved on average in the function optimization problem.

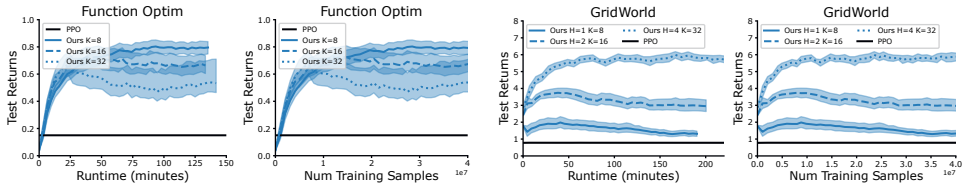


Figure 4.3: Learning curves for all environments comparing our VariBASeD against recurrent PPO (RL^2) (Duan et al., 2016) using the S5 architecture (Lu et al., 2023) for different planning budgets. Shaded regions give 99% two-sided BCa-bootstrap intervals over 30 seeds. In our framework we found that the PPO baseline did not learn.

Learning Curves. While Figure 4.2 shows results for single trained agents, we also plot the full learning curves over multiple training repetitions in Figure 4.3. Specifically we show how the learning curves are affected by planning budget. On the function optimization problem we used a search-depth of $H = 1$, with a varying particle budget of $K \in \{8, 16, 32\}$. In contrast, on the gridworld, we used a coupled planning budget of $\langle H = 1, K = 8 \rangle$, $\langle H = 2, K = 16 \rangle$, and $\langle H = 4, K = 32 \rangle$. We included the PPO ($RL^2$) baseline for reference; however, we found that it could not effectively learn in our specific setup.

Interestingly, we find an opposing pattern on each environment when comparing the effect of the planner budget. As hypothesized, on the gridworld problem, we find that increasing planner budget significantly improves the agent's efficacy. In contrast, increasing planner budget slightly reduced agent performance on the function optimization problem, which goes against our hypothesis. Furthermore, we also find that all learning curves quickly level-off and begin to slightly deteriorate as learning continues. So, although this finding supports our second claim, the impact of this result is limited if lower budgets prevent the policy from reaching optimality.

Discussion. We hypothesize that our VariBASeD agent does not enjoy much from increased planner budget on the function optimization problem due to an additional exploration effect introduced by using fewer particles. The smaller particle budget adds noise into the overall learning process, which induces more diverse data collection and prevents premature convergence to a suboptimal policy.

This pattern does not hold on the gridworld problem which strongly benefits from additional budget. In contrast to the function optimization, the gridworld environment has actual state-dynamics, and additional lookahead enables the planner to make better use of the value and reward models. We expect that the gridworld requires additional budget for accurate policy inference, whereas for function optimization, a lower budget was sufficient, and the increased budget likely resulted in overly greedy action selection.

Overall, we suspect that the complex loss landscape and caching of the recurrent neural network state causes a representational misalignment that impedes effective learning with our current architecture. This hypothesis implies that simulation based inference is crucial for enabling good performance in our framework. This routine corrects the mispredictions from the neural networks through a proper inference step while still leveraging the efficiency benefits of having an approximate guide during search. If this is true, that would

explain the poor result obtained by our PPO agent, since the loss landscape could be too difficult for gradient descent to navigate for obtaining a proper policy improvement. At the same time, PPO's performance could also be attributed to poor tuning, which resulted in a too difficult optimization landscape. With that said, our approach shows a simpler yet effective way to obtain policy improvement, by scaling up the planner budget over meticulous parameter tuning.

4.5 Related Work

Belief Learning. Much prior work has leveraged sequence models to perform approximate belief learning in RL (Becker et al., 2024, Gregor et al., 2019, Igl et al., 2018, Rakelly et al., 2019). Most relevantly to us, Lu et al. (2023) adapt the S5 sequence model (J. T. Smith et al., 2023) to RL by modifying its architecture to handle episode termination in the hidden state. State-space models outperform traditional RNNs for long-horizon problems, due to the associativity of their hidden state dynamics, which enables training-time parallelism while preserving efficient inference on the GPU. Also, in contrast to transformers (Parisotto & Salakhutdinov, 2021), S5 models can truncate histories during inference by storing their hidden state between batches. This drastically improves their runtime efficiency.

Our approach builds on this framework, where we in essence add regularization and reparameterization to the hidden state via policy-conditioned trajectories. Within this space, VariBAD separately trains a belief model for BA-MDPs by maximizing an evidence lower bound conditional on the current policy (Zintgraf et al., 2021), after updating the belief model, they then separately train the policy conditional on this learned belief (akin to coordinate ascent). Our approach differs from VariBAD in two ways: 1) we treat the policy as an actual part of the probabilistic graphical model, resulting in a natural policy regularization term for the belief-state, and 2) our learning objective is tailored for joint end-to-end optimization of both the policy and belief, our loss scales linearly with the trajectory length in contrast to VariBAD's quadratic scaling.

Similar to our work, Jeong et al. (2025) recently built on the VariBAD framework to add online planning through a type of random-shooting. However, their method does not evolve the posterior throughout planning and further inherits VariBAD's pathologies, ours does not.

Also related to this, RL^2 (Duan et al., 2016) and PEARL (Rakelly et al., 2019) encode past experience using a deterministic RNN or a Neural Process, respectively. These approaches only support the learning of the value and policy, which doesn't produce proper belief states as they are independent of the transition and reward model. Such "improper" beliefs are also often encountered in decision-focused learning (Abdulsamad et al., 2025, Lacoste–Julien et al., 2011, Schrittwieser et al., 2020), our framework accommodates both approaches to enable online planning (Guez et al., 2019).

SMC Planning. Sequential Monte Carlo (SMC) methods for planning have been applied in multiple contexts, including reinforcement learning (Macfarlane et al., 2024), steering large language models (Lew et al., 2023), and Bayesian experimental design (Iqbal et al., 2024). Unlike traditional particle filters focused on state estimation, these approaches lever-

age SMC for inferring an optimal regularized policy. Crucially, they leverage variational inference (Naesseth et al., 2018) to learn a regularized sampling strategy to boost efficacy of the SMC-planner.

The canonical variational SMC, however, requires taking gradients of the SMC procedure (Naesseth et al., 2018). This is problematic due to non-differentiable resampling. Later work, however, showed that this is unnecessary for control, using Fisher's identity to avoid differentiation altogether (Abdulsamad et al., 2025).This motivated a REINFORCE-like objective, and enables efficient policy learning without directly differentiating through the particle system.

In parallel, S. Gu et al. (2015) propose using SMC to sample approximate solutions to the optimal variational distribution. This approach enables using SMC as a probabilistic planner (Piché et al., 2019). Macfarlane et al. (2024) then used this planner in an EM-style learning framework for RL (Abdolmaleki et al., 2018). More recent methods enhance SMC planning even further by improving statistic estimation inside the planner (de Vries, He, Oren, & Spaan, 2025). Our method is also framed around EM, using SMC style planning for generating actions and policy learning targets while using a learned variational belief model. In contrast to prior SMC-based planners, our use of a learned belief model introduces a model mismatch between the proposal and the generative dynamics. We address this by adapting the importance sampling weights to account for model inaccuracies.

4.6 Conclusion

We presented VariBASeD, a scalable and holistic framework for approximate Bayes-adaptive planning that combines sequential Monte-Carlo inference with amortized variational learning of belief states. By coupling online SMC planning with gradient-based belief optimization, our method is posed to improve both sample and runtime efficiency over model-free baselines. We hypothesized that this is due to extracting more informative learning target per inference step, which ends up accelerating the entire training pipeline. Preliminary experiments indeed demonstrate that shifting computational effort from training to online planning yields improved performance in small scale meta-reinforcement learning settings. Future work will investigate domains with longer context lengths and higher task variety to demonstrate the potency of our method on large-scale adaptive control.

Appendix

4.A Derivations & Proofs

Proof 4.A.1. (Theorem 4.1) We want to confirm whether $q^* = \arg\max_q \mathcal{J}_{proxy}(q, \pi^+)$ is consistent with $p^+_\pi(A_t|S_t, b_t, \mathcal{O}_{t:T})$. If we write this in terms of the full trajectory distribution, we get,

$$\min_q KL(p_q(H_t) \| p^+_\pi(H_t|\mathcal{O}_{t:T})). \tag{4.12}$$

Notably, we have extended the states S_t with the belief state b_t to obtain an augmented state-space $\langle S_t, b_t \rangle \in \mathcal{S} \times \mathcal{B}$. If we then assume that the augmented dynamics and rewards

follow the definition stated in the background,

$$p_S^+(b_{t+1}, S_{t+1}|S_t, A_t, b_t) = \delta(b_{t+1} = p(M_{t+1}|H_{<t} \cup H_t))\mathbb{E}_{M_t \sim b_t} p(S_{t+1}|A_t, S_t, M_t),$$

$$p_R^+(R_t|S_t, A_t, b_t) = \mathbb{E}_{M_t \sim b_t} p(R_t|A_t, S_t, M_t), \quad \pi^+(A_t|S_t, b_t) = \mathbb{E}_{M_t \sim b_t} \pi(A_t|S_t, M_t).$$

Then the stated Theorem reduces to a reformulation of Theorem 3.1 in Chapter 3 with a modified reward and transition density in the graphical model. Thus, the proof follows identically and we can safely state that $q^*(A_t|S_t, b_t) = p_\pi^+(A_t|S_t, b_t, \mathcal{O}_{t:T})$. □

Proof 4.A.2. (Proposition 4.2) Similar to Theorem 4.1, this is a reformulation of the dynamics and reward function of the probabilistic graphical model discussed in Chapter 3. Similarly, the proof can be found in Proposition 3.A.3. □

Proof 4.A.3. (Corollary 4.3) Using the target definition from Eq. (4.3) and the imposed structure by Theorem 4.1, we first manipulate the marginal density $p_\pi(H_{1:t})$ to obtain a weight-recursion in terms of history dependence,

$$\mathbb{E}_{p_\pi(H_{1:t}|\mathcal{O}_{1:T})} f(H_{1:t}) = \mathbb{E}_{p_q(H_{1:t})} \frac{p_\pi(H_{1:t}|\mathcal{O}_{1:T})}{p_q(H_{1:t})} f(H_{1:t}) = \mathbb{E}_{p_q(H_{1:t})}[w_t \cdot f(H_{1:t})] \tag{4.13}$$

$$= \mathbb{E}_{p_q(H_{1:t})} \left[\frac{p_\pi(H_t|H_{1:t-1}, \mathcal{O}_{t:T}) p_\pi(H_{1:t-1}|\mathcal{O}_{1:T})}{p_q(H_{t|H_{1:t-1}}) p_q(H_{1:t-1})} f(H_{1:t}) \right] \tag{4.14}$$

$$= \mathbb{E}_{p_q(H_{1:t})} \left[w_{t-1} \frac{p_\pi(H_t|H_{1:t-1}, \mathcal{O}_{t:T})}{p_q(H_{t|H_{1:t-1}})} f(H_{1:t}) \right] \tag{4.15}$$

$$= \mathbb{E}_{p_q(H_{1:t-1})} \left(w_{t-1} \mathbb{E}_{p_q(H_t|H_{1:t-1})} \left[\frac{p_\pi(H_t|H_{1:t-1}, \mathcal{O}_{t:T})}{p_q(H_t|H_{1:t-1})} f(H_{1:t}) \right] \right) \tag{4.16}$$

Now extract the w_t term for brevity and separate the conditioning on $\mathcal{O}$,

$$w_t = w_{t-1} \cdot \mathbb{E}_{p_q(H_t|H_{1:t-1})} \left[\frac{p_\pi(S_{t+1}, R_t, A_t|H_{1:t-1}, \mathcal{O}_{t:T})}{p_q(S_{t+1}, R_t, A_t|H_{1:t-1})} \cdot f(H_{1:t}) \right] \tag{4.17}$$

$$= w_{t-1} \cdot \mathbb{E}_{p_q} \left[\frac{p_\pi(S_{t+1}, R_t, A_t|H_{1:t-1})}{p_q(S_{t+1}, R_t, A_t|H_{1:t-1})} \cdot \frac{p_\pi(\mathcal{O}_{t:T}|S_{t+1}, R_t, A_t, H_{1:t-1})}{p_\pi(\mathcal{O}_{t:T}|H_{1:t-1})} \cdot f(H_{1:t}) \right] \tag{4.18}$$

$$= w_{t-1} \cdot \mathbb{E}_{p_q} \left[\frac{p_\pi(S_{t+1}, R_t, A_t|H_{1:t-1})}{p_q(S_{t+1}, R_t, A_t|H_{1:t-1})} \cdot \frac{p(\mathcal{O}_t|R_t) p_\pi(\mathcal{O}_{t+1:T}|H_{1:t})}{p_\pi(\mathcal{O}_{t:T}|H_{1:t-1})} \cdot f(H_{1:t}) \right] \tag{4.19}$$

$$\propto w_{t-1} \cdot \mathbb{E}_{p_q} \left[\frac{p_\pi(S_{t+1}, R_t, A_t|H_{1:t-1})}{p_q(S_{t+1}, R_t, A_t|H_{1:t-1})} \cdot \exp R_t \cdot \frac{\exp V_{soft}^{q,\pi}(H_{1:t})}{\exp V_{soft}^{q,\pi}(H_{1:t-1})} \cdot f(H_{1:t}) \right] \tag{4.20}$$

where the proportionality is due to $p(\mathcal{O}_t|R_t) \propto \exp R_t$ and where $V_{soft}^{q,\pi}(H_{1:t}) = \ln p_\pi(\mathcal{O}_{t+1:T}|H_{1:t})$. For brevity, assume that $p(\mathcal{O}_t|R_t) = \exp R_t$ properly normalizes as is from this point on.

Now to deal with history-dependence, we will insert the true and variational belief states,

$$\frac{w_t}{w_{t-1}} = \mathbb{E}_{p_q(H_t|S_t, b_t^q)} \left[\frac{p_\pi(S_{t+1}, R_t, A_t|S_t, b_t)}{p_q(S_{t+1}, R_t, A_t|S_t, b_t^q)} \cdot \exp R_t \cdot \frac{\exp V_{soft}^{q,\pi}(S_{t+1}, b_{t+1})}{\exp V_{soft}^{q,\pi}(S_t, b_t)} \cdot f(H_{1:t}) \right]. \tag{4.21}$$

Observe that $\mathbb{E}_{p_q(H_t|S_t,b_t^q)}(\cdot) = \mathbb{E}_{\mathbb{E}_{M\sim b_t^q} p_q(H_t|S_t,M)}(\cdot)$ per definition (see Background), this can be rewritten more simply as $\mathbb{E}_{b_t^q(M)}\mathbb{E}_{p_q(H_t|S_t,b_t^q)}(\cdot)$ using Fubini's theorem.

Reparametrize the value functions $V^{q,\pi}_{soft}(H_{1:t-1}) = V^{q,\pi}_{soft}(S_t, b_t) = \mathbb{E}_{M\sim b_t} V^{q,\pi}_{soft}(S_t, M)$ to obtain,

$$\frac{w_t}{w_{t-1}} = \mathbb{E}_{b_t^q}\mathbb{E}_{p_q}\left[\frac{\mathbb{E}_{b_t} p_\pi(S_{t+1}, R_t, A_t|S_t, M_t)}{\mathbb{E}_{b_t^q} p_q(S_{t+1}, R_t, A_t|S_t, M_t)} \exp R_t \frac{\mathbb{E}_{b_{t+1}} \exp V^{q,\pi}_{soft}(S_{t+1}, M_{t+1})}{\mathbb{E}_{b_t} \exp V^{q,\pi}_{soft}(S_t, M_t)} f(H_{1:t})\right]. \tag{4.22}$$

Importance sampling with b_t^q, s.t., $\omega_t = \frac{b_t(M)}{b_t^q(M)}$ for all terms involving $\mathbb{E}_{b_t}(\cdot)$ obtains the result. □

Proof 4.A.4. (Proposition 4.4) The first part of the stated lower-bound (optimization objective) restates the property of Proposition 4.1, which only leaves the right-term. The KL-divergence has a well known decomposition into an lower-bound and evidence term,

$$\begin{aligned}
\mathcal{L}_{\mathcal{Z}}(b_{t+1}^q) = KL(b_t^q \| b_t) &= KL(q(M_t|H_{<t}) \| p(M_t|H_{<t})) && (4.23)\\
&= \mathbb{E}_q[\ln q(M_t|H_{<t}) - \ln p(M_t|H_{<t})] && (4.24)\\
&= \mathbb{E}_q[\ln q(M_t|H_{<t}) - \ln p(M_t, H_{<t}) + \ln p(H_{<t})] && (4.25)\\
&= \ln p(H_{<t}) + \mathbb{E}_q[\ln q(M_t|H_{<t}) - \ln p(M_t, H_{<t})] && (4.26)\\
&= \ln p(H_{<t}) + \mathbb{E}_q[\ln q(M_t|H_{<t}) - \ln p(H_{<t}|M_t) - \ln p(M_t)] && (4.27)\\
&= \ln p(H_{<t}) - \mathbb{E}_q[\ln p(H_{<t}|M_t)] + KL[q(M_t|H_{<t}) \| p(M_t)] && (4.28)\\
&= \ln p(H_{<t}) - \mathcal{L}_{\mathcal{Z}}(b_t^q) && (4.29)
\end{aligned}$$

where the right term is another ELBO,

$$\begin{aligned}
\mathcal{L}_{\mathcal{Z}}(b_{t+1}^q) = \sum_{i=1}^{t} \mathbb{E}_{b_{t+1}^q}[&\ln p(S_{t+1}|S_t, A_t, M_t) + \ln p(R_t|S_t, A_t, M_t)\\
&+ \ln \pi(A_t|S_t, M_t)] - KL(b_{t+1}^q \| b_0),
\end{aligned} \tag{4.30}$$

which can be read out from Eq. (4.1). Finally, observe $\max_q \Omega^t_{\mathcal{Z}} = \max_q \mathcal{L}^{\mathcal{Z}}(q)$ as the log-evidence is a constant. □

4.B Control as Inference Details

4.B.1 On Determinstic Rewards in Control-as-Inference

The probabilistic graphical model we consider in the main text (Figure 4.1) assumes a deterministic reward function to prevent maximizing behavior over noise. Although the Control-as-Inference (Levine, 2018, Ziebart, 2010) does not prevent us from working with stochastic rewards, we should define the outcome variables $\mathcal{O}_t$ to only depend on the mean reward for parity with reinforcement learning objectives.

To see why, assume an MDP setting where we have μ_{R_t} the mean of the reward function and a noisy realization of R_t. The inverse problem posed by $p(\mathcal{O}_t|A_t, S_t)$ can be expressed

as $\frac{1}{\mathcal{Z}} \int p(\mathcal{O}_t|R_t)p(R_t|S_t,A_t)dR_t$ in the stochastic setting, and simply as $\frac{1}{\mathcal{Z}} \int p(\mathcal{O}_t|R_t)\delta(R_t - \mu_R(S_t,A_t))dR_t = \frac{1}{\mathcal{Z}} p(\mathcal{O}_t|\mu_R(S_t,A_t)) = \frac{1}{\mathcal{Z}} \exp \mu_R(S_t,A_t)$ in the deterministic setting (with $\mathcal{Z}$ the normalizing constant). Suppose the noisy rewards were Gaussian distributed $R_t \sim \mathcal{N}(R_t, \mu_{R_t}, \sigma_R^2)$, then the integration over R_t is simply given by the moment generating function $(\mathcal{O}_t|A_t,S_t) = \frac{1}{\mathcal{Z}} \exp(\mu_{R_t} + \sigma_R^2/2)$. Intuitively, when evaluating $p(\mathcal{O}_t|A_t)$ over the entire action space $\mathcal{A}|S_t$, the role of the exponential function is to then smoothly shift probability density to the actions that have higher rewards, and shrink density from actions with low rewards. In other words, this example shows a softmax (Boltzmann) policy based on the rewards. Thus, naively using noisy rewards in the Control-as-Inference framework leads to behavior that seeks out noisy rewards, and using a mean dependency for $\mathcal{O}_t$ does not.

This problem can be easily remedied though, namely by using a model to learn the average rewards. In our setting, this means that SMC-planning should use the average-reward prediction (and not samples) to compute importance-sampling weights, as discussed in Section 4.3.1. Furthermore, we should also use value estimators and learning in the outer-loop such that this is consistent with the expected returns, which is almost always the case, except in the subtle case of some distributional settings (Bellemare et al., 2017).

4.C Experiment Details

4.C.1 Environments

For the simple discrete environment setup, we used our own implementation of a gridworld. This simple gridworld is almost equivalent to the implementation by Zintgraf et al. (2021), where upon resetting the environment, a random goal and start tile are drawn in an open-grid. The agent must then explore each tile to find the hidden goal-tile, the agent can explore over multiple episodes each with a strict timelimit.

For the simple continuous control, we used a continuous function optimization task similar to Bayesian optimization methods (Y. Chen et al., 2017). This environment is stateless, similar to the Bandit setting, however the belief-state is of course maintained over multiple interactions (episodes). Upon resetting the environment, we generate a new function for the agent to optimize.

Continuous Function optimization environment details (ours):

- The observation is the output value of the n-dimensional function given the previous input.
- The reward is the average over all function dimensions for the given input, $r_t = \frac{1}{D} \sum_{i=1}^{D} f_i(x_{t,i})$.
- The agent can interact with the function for 20 total steps. Although the problem is stationary, it means that the belief-state horizon is 20 steps.
- Upon a reset, new functions $\{f_i\}_{i=1}^{D}$ are generated i.i.d. $\forall f_i$, by randomly sampling the parameters of a $n = 3$ truncated Fourier series. For this, we uniformly sample the amplitudes $a \in [0.1, 1.1]$, phase $\phi \in [0, \pi]$, and an input shift $c \in [0, \pi]$, for each component-wave, such that $f(x) = a_0 + \sum_{i=1}^{n-1} a_i \cos(2\pi(x - c) - \phi)$.

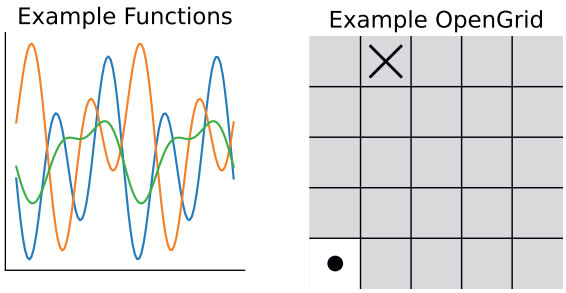


Figure 4.4: Visualization of environments. Left: continuous function optimization task, where each color corresponds to a distinct function that the agent has to learn to optimize from sequential interactions. Right: discrete opengrid task where the agent needs to find the goal tile × over multiple episodes starting from the dot.

Discrete GridWorld environment details (ours):

- The observation is a $n \times n$ image with 1 channel of values between $[0, 1]$, where 0 means an empty tile and 1 means the agent's current position.
- The action space is a discrete choice of moving up, down, left, right, or staying still. If the agent moves against a wall, it remains on the same tile.
- The reward is zero everywhere, except for a reward of $\frac{1}{\text{time}}$ when the agent is on the goal tile.
- The agent terminates after 10 environment steps. The environment is repeated for 6 episodes in total before resetting.
- Upon a reset, the goal and start tile are randomly initialized somewhere in the grid.

4.C.2 Hardware Requirements

All experiments were run on a GPU cluster with a mix of NVIDIA Tesla V100-SXM2 32GB, NVIDIA A40 48GB, and A100 80GB GPU cards. Each run (random seed/repetition) required only a few CPU cores (2 logical cores) with a low memory budget (e.g., 4GB). We also tested locally on a NVIDIA GeForce RTX 4080, for which most agents could be trained in a few hours. Thus, our implementation is readily applicable on consumer grade hardware.

4.C.3 Hyperparameters, Model Training, and Software Versioning

The hyperparameters for our experiments are summarized in Table 4.1. Note that all parameter values indicated in sets were run in an exhaustive grid for the ablations.

Given our final loss function $\mathcal{L}(\theta, \mathcal{D})$, recall that the E-step is completed for the variational belief $b^q_{\phi_t}$ by doing gradient-descent on the loss to produce $\theta'_{(n)} = \theta_{(n)} - \alpha \nabla \mathcal{L}(\theta)|_{\theta=\theta_{(n)}}$, its M-step is then performed as $\theta_{(n+1)} \leftarrow \theta'_{(n)}$. This mixes together the M-step for the policy,

whose E-step we already performed during data-generation. In other words, the E-step for the variational belief, is mixed together with the M-step for the policy/value network.

We optimize $\mathcal{L}$ with stochastic gradient descent (see the hyperparameter tables) using the AdamW optimizer (Loshchilov & Hutter, 2019) with an l_2 penalty of 10^{-6} and a learning rate of $3 \cdot 10^{-3}$. Gradients were clipped using two methods, in order: a max absolute value of 1 and a global norm limit of 1.

The replay buffer was implemented as a uniform circular buffer with size:

$$\text{max-age of data} \times \text{number of parallel environments} \times \text{number of unroll steps}.$$

The max-age of data was tuned to fit into reasonable GPU memory.

Additionally, Table 4.2 provides version information for key software packages. We implemented everything based on Jax 0.4.30 in Python 3.12.4.

4.C.4 Neural Network Architectures

Continuous Function Environment

- Embedding: 3-layer MLP with 128, 64, 32 nodes per layer for the action, observation, and reward separately.
- Leaky-ReLU activations followed by LayerNorm.
- S5 architecture with 4 layers and Layernorm post-transformation. Other parameters follow the defaults of Lu et al. (2023).
- The hidden-state of the S5 is projected to a 32 dimensional Gaussian with independent variances. We use 10 samples for the ELBO through reparametrization to obtain multimodal predictive distributions.
- Policy, value, and reward networks all use a 3-layer MLP with 128, 64, 32 nodes per layer.
- For the policy, the last layer is linearly projected to parametrize a diagonal multivariate Gaussian squashed via Tanh, as in Haarnoja et al. (2018).
- For the value and reward, the last layer is projected to a scalar to parametrize the mean of a univariate Gaussian.

Discrete Grid Environment

- Embedding: 3-layer MLP with 128, 64, 32 nodes per layer for the action and reward separately.
- Embedding: 2-layer CNN with 4 channels, a stride of 2, and a 3 by 3 kernel for the grid-observation.
- Leaky-ReLU activations followed by LayerNorm.

Table 4.1: Experiment hyperparameters. Values in sets were varied as part of our ablations.

Name	Symbol	Value Function Env	Value Grid
SGD Minibatch size		1024	1024
SGD update steps		32	32
Unroll length (nr. steps in env)		64	128
Batch-Size (nr. parallel envs)		32	32
(outer-loop) TD-Lambda	λ	1.0	1.0
(outer-loop) Discount	γ	0.99	0.99
Value Loss Scale	c_v	0.5	0.5
Policy Loss Scale	c_π	1.0	1.0
Policy Entropy Loss Scale	c_{ent}	0.0003	0.1
Loss belief-state burn-in length		8	12
Loss decode window size		4	6
Loss unroll window size		4	6
Belief KL-penalty		0.01	0.01
Belief Entropy Penalty		10^{-5}	10^{-5}
PPO			
PPO Policy-Ratio Clipping	ϵ	0.3	0.3
PPO Value loss Reward Scale		10	10
SMC			
Replay Buffer max-age		16	16
Planner Depth	m	$\{1, 2, 4, 6\}$	$\{1, 2, 4, 6\}$
Number of particles	K	$\{8, 16, 32, 64\}$	$\{8, 16, 32, 64\}$
Resampling period	r	2	2
Target temperature	T	0.1	0.1

Table 4.2: Software module versioning that we used for our experiments (also includes default parameter settings).

Package	Version
brax	0.10.5
optax	0.2.3
flashbax	0.1.2
rlax	0.1.6
flax	0.8.4
xland-minigrid	0.9.1

- S5 architecture with 4 layers and Layernorm post-transformation. Other parameters follow the defaults of Lu et al. (2023).
- The hidden-state of the S5 is projected to a 32 dimensional Gaussian with independent variances. We use 10 samples for the ELBO through reparametrization to obtain multimodal predictive distributions.
- Policy, value, reward, and state networks all use a 3-layer MLP with 128, 64, 32 nodes per layer.
- For the policy, the last layer is linearly projected to parametrize the logits.
- For the value and reward, the last layer is projected to a scalar to parametrize the mean of a univariate Gaussian.
- For the state prediction network, the last layer is projected to independent Bernoulli probabilities for each tile in the observed grid. This probability is a transition probability to predict where the agent moves towards.

5

Discussion

A society grows great when old men plant trees in whose shade they shall never sit.

– Greek proverb

This dissertation set out to present a joint vision on Bayes optimal decision making in the context of optimal experiment design in an autonomous laboratory. In this chapter we aim to contextualize our findings in a broader perspective. The content presented in Chapters 2 to 4 provides distinct contributions to the machine learning field, we summarize each of these and link them back to the **research questions (RQs)** outlined in Chapter 1. We simultaneously reflect on how this contrasts with Bayesian optimization, to see how moving to deep **reinforcement learning (RL)** introduces unique (or overlapping) challenges, and that our contributions make a step towards improving these issues. Afterwards we provide a distinct outlook for the technical and applied side of current **artificial intelligence (AI)** methodologies, followed by closing remarks to emphasize a future vision on general **AI** in embodied systems.

5.1 Research Findings

Answer to RQ 1: *"Can we capture optimal sequential decision-making in a probabilistic model that generalizes Bayesian optimization and reinforcement learning?"*

For the latter part of this **RQ**, this is answered in the affirmative in Chapters 2 & 4 through the control-as-inference framework (Levine, 2018, Ziebart, 2010). We also show that meta-learning with deep neural networks allows for scalable approximation of this model through learning/amortization. For Bayes-optimal behavior, our formulation extends upon prior work in Bayes-adaptive **Markov decision processes (MDPs)** for deep reinforcement learning by Zintgraf et al. (2021). The crucial difference is that our formulation embeds the control component as part of the environment modeling, rather than seeing it as

separate. This presents a straightforward way to perform regularized approximate policy iteration. One consequence however, is that our formulation overemphasizes learning 'local' environment features for approximating uncertainty (the belief state) rather than capturing the global uncertainty as done by VariBAD (Zintgraf et al., 2021). We provide a further discussion on how to circumvent this shortcoming in the future work section.

Although this dissertation has not focused much on the methodological side of the **Bayesian optimization (BO)** field, it remains useful to complete the link to **RL** due to its widespread adoption in practice (as discussed in Chapter 1). There is a simple and well-known connection to **RL** that makes **BO** a subfield of methods which we reiterate within our framework. The fundamental difference distinguishing **BO** from **RL**, when viewed from our probabilistic model discussed in Chapter 4 (see Figure 4.1), is that **BO** applies the simplifying assumptions:

1. The decision events are independent of one another given their current belief state.
2. Typically, the environment dynamics are unchanging, independent of previous events, or depend on an affine transformation of the previous system state (Kalman filtering; Kalman, 1960).

There should be an emphasis on 'typically', since the literature has plethora of variations that all handle specific domain characteristics (as we cautioned for in the Chapter 1). In simple terms, the first item implies that decisions that our agent takes in the present do not need to account for what happens afterwards: they are short-sighted[1]. In contrast, **RL** tackles decision taking holistically by accounting for the entire future horizon. Then, for the second item, this is a typical characteristic to maintain simplicity in the assumed uncertainty modeling, i.e., the belief state, such that it preserves efficient closed-form computation. As a result, uncertainty can be handled exactly, which simplifies approximation strategies considerably, so long as the user can specify a proper prior and model likelihood. The combination of linear likelihoods, Gaussianity of the belief state, and the first assumption drastically simplifies decision taking, often allowing for closed form solutions on the expected return under the posterior (like the upper-confidence bound, or probability of improvement).

The final step in making the connection of our model in Figure 4.1 is that **BO** usually does not require approximate policy iteration. Instead, it maximizes the expected return through explicit online search. Intuitively, this can be understood as generating the most probable path (the mode) within our probabilistic graphical model (Bishop, 2007). One particular downside of this approach is that this explicit search can be expensive in large input spaces (e.g., combinatorial search spaces). However, ideas from **RL** and **BO** can be combined of course to make this more manageable; such as amortization of search for warm-starting (González-Duque et al., 2024b), or using dynamic programming to fill in a batch of actions (see Example 1.4 from Chapter 1).

To conclude, we can also answer **RQ** 1 in the affirmative for **BO**. This is crucial for making statements about the approximations and methods we develop for **RL** in an alternative

[1]Again this also needs to be contextualized, since careful construction of exploration strategies might appear myopic in the graphical model formulation, whereas they are actually infinitely farsighted. The upper-confidence bound is such a heuristic (Munos, 2014), but entropy-search is not (Hennig & Schuler, 2012).

BO setting. Since **BO** is a popular approach in experiment design (Roch et al., 2020), this connection should give stronger insight into what reasons might support or caution against a transition from **BO** to the more general deep **RL** setting. To this end we suggest the following: while **BO** could imply a misspecified model for an optimal experiment-design agent (depending on the use-case), its simplifications can allow it to push past the performance of 'well specified' **RL** agents due the difficulties in obtaining good approximations to said **RL** agent. Nevertheless, pushing the frontiers in scaling Bayesian **RL** agents is worthwhile in the long term, as theoretically it prevents short-sightedness in exploration. Naturally, in practice it is wise to balance the long-term prospects of **RL** with the benefits of simpler existing approaches like **BO** in the short-term, which ultimately boils down to a value judgment of the designer on a case-by-case basis.

Answer to RQ 2: *"How do contemporary deep reinforcement learning algorithms fit within our framework? What can we say about the quality of their approximations?"*

Firstly, Chapter 2 showed how meta-**RL** can be understood as an amortization approach to an underlying probabilistic graphical model with an uncertain (latent) **MDP** variable. Subsequently, Chapters 3 & 4 show that we can formulate online planning style algorithms, like AlphaZero which uses **Monte-Carlo tree search (MCTS)** or our **sequential Monte-Carlo (SMC)** planner, through an **expectation-maximization (EM)**-loop, a classic probabilistic estimation method. Although prior work by Grill et al. (2020) point this out too, we extend this formulation further to the uncertain **MDP** case. This observation corroborates a nice connection between (Bayes-) optimal decision making and straightforward probabilistic inference.

Subsequently, to reiterate our conclusion from Chapter 2, we find that conventional approaches like RL^2 (Duan et al., 2016) typically do not approximate uncertainty particularly well. At least, not when understood as approximating a posterior distribution over a (latent) **MDP** measure. Methods like RL^2 use point-estimates to amortize inference to this posterior, which we show is not just statistically inconsistent (Xiong et al., 2021), but also tends to become overconfident as measured by the posterior differential entropy. It is unsurprising that maintaining statistical consistency requires some degree of online finetuning, as also confirmed by Xiong et al. (2021). However, the question of "how to learn a representation that is sufficiently general for efficient online adaption" (Finn et al., 2017) remains open. We stipulate directions for future work on this matter in the following section.

Although our findings in Chapter 2 cautioned us to perform proper variational modeling of the latent posterior in Chapter 4, we still find that this does not truly capture the agent's uncertainty. In our VariBASeD algorithm, we constructed a belief learning loss that trains on mostly local data. In contrast, prior work by Zintgraf et al. (2021) instead tries to capture global uncertainty by predicting entire trajectories. Although predicting full trajectories does not scale well with increasing trajectory length, this finding emphasizes that amortization of the belief needs complete information to more accurately capture "global" uncertainty. Thus, when we train belief states through past, current, or future token prediction, we can really only expect it to learn features that support this task and nothing else, except for the serendipitous case when our model generalizes. This is an unfortunate lesson since tractable methods should learn from streams of experience, and

should not have to remember complete datasets for inference. In the future work section we also sketch some approaches for how we could develop methods that capture global uncertainty from local streams of information.

Answer to RQ 3: *"How do we efficiently scale approximate inference within our model formulation?"*

There has been a relatively well accepted consensus that learning and planning (online inference) are both a promising approach to scale inference to intractable decision making problems (Sutton & Barto, 2018). This was also a philosophy we adopted through development of our methods, it is how we scaled up posterior inference in Chapters 2 & 4. Furthermore, we similarly approached combining policy learning and planning in Chapters 3 & 4. This might sketch the idea that we already knew how to ideally scale up inference to Bayes-optimal behavior, however, as discussed in Chapter 3 the development of algorithms that can maximally make use of their compute resources is still at the frontier of research. Prior approaches based on **MCTS** had shown impressive performance in boardgames, algorithm discovery, and more (Mankowitz et al., 2023a, Silver et al., 2018), yet **MCTS** has remained difficult to scale up even further due to its sequential nature. Our contributed **SMC** planner for **RL** showed that we can match or surpass data-efficiency compared to **MCTS**, but also improve upon algorithm runtime compared to both **MCTS** and non-planning based methods.

This latter finding presents a somewhat counterintuitive insight, online inference or planning during an agent's training phase not only improves data-efficiency but also its overall runtime. It is counterintuitive because doing online inference requires more computation, and also runtime, compared to strict amortized inference. However this ignores that the agent's training cycle includes parameter updates through a **stochastic gradient-descent (SGD)** cycle. The key insight is that performing online inference improves the agent's accuracy during data-collection, on which it will then subsequently learn. This allows our algorithm to shift a major computation bottleneck; if agent-environment interaction is comparatively 'cheap' and **SGD** is expensive, we can shift some of our **SGD** budget to online inference at negligible cost to significantly improve overall training runtime.

Note that this is mostly an empirical finding. For instance, we do not provide any theoretically substantiated compute-optimal budgets for planning and learning in this setting. In other words, we conjecture a trade-off in optimally scaling up planning budget proportional to the learning/weight-update budget. Furthermore, our setting is also restricted in scope, we do not consider extending compute on offline data (e.g., using reanalyze mechanism; Schrittwieser et al., 2020) or how this scaling pattern holds when performing full world-model learning (Hafner et al., 2020). Nevertheless, our insight could prove useful for inspiring future work.

5.2 Outlook

Based on the contributions outlined in the previous section, it is important to zoom out and position our work in a broader context. This section lays out what we perceive as the most pressing bottlenecks and limitations for training deep reinforcement learning methods for

autonomous decision-making under uncertainty. The first part presents a more technical focus, while the second part links back to the goal of autonomous laboratories discussed in Chapter 1.

5.2.1 Limitations and Future Work

Belief Learning. As affirmed in Chapter 2, and improved in Chapter 4, the framework we adopted for modeling task uncertainty is mostly **local**. That is, the representations and features captured by our variational belief learning algorithm tend to overemphasize recent information at the expense of the distant past and future. Consequently, such learned models cannot be expected to capture an agent's true uncertainty (Zintgraf et al., 2021). This is not surprising, as we explicitly formulate the loss function on local subtrajectories that also only predict local data. Unfortunately, learning from strictly local data is a crucial feature of an algorithm in order to achieve tractable online learning (Silver & Sutton, 2025).

As observed in Chapter 4, using gradients for optimization from reinforcement learning losses, like policies and values, improves the agent's performance by distilling global information (i.e., the future) into the learned belief. This has both positive and negative effects. Firstly, it enables abstraction based on supporting optimal behavior, which often proves useful in approximate settings with limited model capacity (Lacoste–Julien et al., 2011, Schrittwieser et al., 2020). This approach reduces variance by tilting the belief state towards features correlated with high expected returns (Abdulsamad et al., 2025). However, a downside is that this does not recover the true belief state (i.e., the true posterior distribution over latent **MDPs**). As a consequence, learned beliefs can become overoptimistic toward tasks within the training distribution rather than forming a general belief which can be reused in downstream problems (Levine, 2018).

We thus identify this as a key limitation and opportunity for future work: designing scalable approaches for approximating belief states that can distill global information from local data. One suggested approach can borrow ideas from the RL loss, which captures global information about the future. This is the forward-backward representation (Touati & Ollivier, 2021), a combination of successor and predecessor features (Barreto et al., 2017, van Hasselt et al., 2021). The latter captures the global information of the past, which is what we desire. Furthermore, since the forward-backward representation is policy-focused and not reward-focused per se, the learned features become more reward agnostic. In turn, this can boost out-of-distribution performance to unseen rewards through an improved generality of the underlying method. Another approach to deal with this issue could also include a hybrid amortized and particle-based filter (Abdulsamad et al., 2025). We leave this open as an important avenue for future work.

Curriculum Learning. Perhaps the strongest limitation of the experimental setups in Chapters 2 & 4 is that we did not utilize any form of curriculum learning (Bauer et al., 2023, Bengio et al., 2009) to accelerate model learning. These methods adjust the training distribution throughout the agent's lifetime to, for example, filter out uninformative data, oversample data with high predictive error, or improve diversity (entropy) in the training data. This is typically a crucial element of meta-**RL** systems since such curricula can boost the learning signal of the agent based on its current performance. As a result, they reduce

overall training runtime and compute requirements.

We opted to leave this out for simplicity, i.e., to remove unnecessary moving parts in our algorithms, but in hindsight, this may have objectively reduced the measured efficacy of our methods. Our approach only sampled tasks from a static, manually specified, task prior. However, it can be particularly challenging to specify this in such a way that achieves the desired test-time behavior. For instance, we found that our manual approach caused the agent to exploit non-general regularities in the training distribution at test-time, as seen in the periodic regularity in Figure 4.2 of Chapter 4. This should be accounted for in follow-up work.

Sequential Monte-Carlo planning. In Chapter 3 we significantly boosted the performance of particle-filter planners for use in deep **RL**. However, our contributions only scratch the surface of what can be done in this area. The **SMC** field provides an abundance of approaches to explore further for planning and variance reduction. For instance, adaptive resampling schemes or anchor particles (Chopin & Papaspiliopoulos, 2020) are examples of low-hanging fruit that can boost our method. We argue that making further improvements to **SMC** planners for **RL** is generally worthwhile for scaling up agents to increased compute, as this approach can improve both the data-efficiency and overall training runtime of the learning algorithms.

Arguably, the most pressing limitation of **SMC** planners is that they do not benefit much from depth. When we scale up the number of forward inference calls within our planner, it often does not improve performance significantly. Scaling fixed budget planners to deep search is fundamentally challenging due to the exponentially increasing variance with planning horizon (Liu et al., 2018). It is likely that further integration with **MCTS** methods (Browne et al., 2012) can help here, as these methods tackle this increasing variance with sequential sampling and value function improvement (Oren, de Vries, et al., 2025). This means that initial iterations of the algorithm are conservative with depth, so that additional samples can be taken in consecutive planner iterations with lower variance estimation deep inside the search tree.

Furthermore, in Chapters 3 & 4 we only considered expectation-maximization style learning algorithms for distilling planning results into neural networks. However, typical variational **SMC** approaches often differentiate through the planner, akin to variational Bayes. Initially, this would seem problematic due to the non-differentiable resampling. However, recent work by (Abdulsamad et al., 2025) shows that one can use Fisher's identity to enable efficient differentiation. This shows that the use of clever reparametrization or differentiation methods (Ścibior & Wood, 2021) can reduce significant variance from sampling through gradient-based learning. Whether such gradient-based approaches or our sample based approach offer more stable and efficient learning is another interesting avenue for future work.

5.2.2 Moving to Automated Designs in Real Laboratories

It should be evident by now that the development of general and optimal sequential decision-making agents is still at the frontier of machine learning research, with pressing

shortcomings in performance, brittleness (robustness), scalability, and reliability. Applying such methods to real-world domains, such as self-driving laboratories, introduces a unique set of additional challenges. However, despite the relative infancy of deep **RL** and its posited challenges, we emphasize that in carefully constructed domains **RL** can offer strong benefits over simpler or more established methods like Bayesian optimization. A general holistic agent[2], one that can accumulate vast amounts of experimental data and perform general decision making over any test task, holds the most potential for maximal data efficiency and economic potency. Because of this promised potential, we present below a number of key challenges to overcome to move towards such an agent.

Firstly, a particularly relevant challenge is how to make ideal use of offline data for reuse in downstream tasks: Should such data be embedded in the model's weights through offline learning? Should we only use this data for offline finetuning of pretrained weights? Or should such offline data be treated as a burn-in sequence for amortized belief-state approaches as we used in Chapters 2 and 4. Furthermore, an additional challenge presents itself in modeling the inter-experiment dynamics of an autonomous laboratory and how such dynamics transfer between different laboratories. For instance, in practice an experimentation cycle may have to balance between different experimentation platforms, platforms with changing experiment throughput, multi-fidelity decision making, or varying frequency feedback. Adequately integrating such diverse and typically imbalanced data is an entire challenge in itself, but is essential in accurately modeling the decision making problem.

Furthermore, tangential to the belief learning challenges posed in the previous subsection, another pressing challenge is that of statistical consistency of learned models when used for experiment planning (i.e., as done in Chapter 4). The approach we adopted in this dissertation relies almost entirely on deep learning to amortize the intractable statistical inference and to enable scaling to complex data. However, deep learning is notorious for its inability to continually learn (Dohare et al., 2024). This means, that the latent representation our models learn does not accurately capture the true underlying statistical routine we desire (Xiong et al., 2021). One particularly humorous side-effect is that learned models will decrease in performance even when given highly informative data about the task, if the model was not strictly trained to deal with such "assistance" in an offline training phase. We observed such an effect in Chapter 2 when extending the horizon of our agent at test-time beyond that during the training phase. We surmise that the development of truly statistically consistent architectures and learning algorithms is absolutely essential for applying deep **RL** in practice to ensure adaptation to unforeseen situations—which in real-life are effectively guaranteed.

Of course, this brief summary cannot capture all technicalities encountered when actually applying deep **RL** in an autonomous laboratory. We did not touch upon challenges such as maintaining sufficient Pareto front coverage in multi-objective settings (Daulton et al., 2020, Roijers et al., 2013), how to deal with ordinal (preferential) rewards when the optimization objective is impossible to quantify in a reward function (Ho et al., 2007), irregularly sampled data and decision boundaries (R. T. Q. Chen et al., 2018, Yildiz et al., 2021), learning termination conditions to prematurely stop experiments, and much more.

[2]Recently the field has started to refer to such agents as behavioral foundation models (Sikchi et al., 2025).

Furthermore, this should also include staying up to date with the most powerful generative modeling techniques in the field to capture the rich multi-modal structure of the real world. All-in-all, there is an abundance of directions that can be further explored to improve both the current state of deep **RL**. This is not disjoint from the broader aim of improving the performance of optimal experiment-design algorithms, but mostly pertains to a shift in focus for research agendas.

5.3 Conclusion

Developing general autonomous decision-making systems is an immensely complex endeavor that has the potential to reshape society for the better. This dissertation contextualized this ambition for the self-driving laboratory use-case. This manuscript started out with a short description of classical **design of experiments (DOE)** for *in-vitro* experimentation and how both **BO** and **RL** naturally emerge as methodological solutions to the challenges that arise when extending our agent with the aim of improved data-efficiency. Based on this buildup, we posited that deep **RL** is the current, but also likely to be the future, state-of-the-art approach to tackle autonomous experimentation in high dimensional and heterogeneous design spaces. However, the implementation of deep **RL** to abstract realizations of the self-driving laboratory (i.e., in simulated/toy environments) still posed both fundamental and practical challenges pertaining to robustness, reliability, scalability, and correct model specification. Thus, this dissertation has taken steps towards realizing deep **RL** to the self-driving laboratory by: investigating the reliability of existing approximate **RL** approaches and improving the runtime and data-efficiency with new **RL** algorithms. Ultimately, our findings should inform the design of future deep **RL** algorithms, this significantly pushes the frontier of making deep **RL** algorithms applicable in practice.

Although we identified and proposed solutions for several technical issues, our focus only scratches the surface of the shortcomings of existing **AI** technology. For instance, we did not explore offline learning to efficiently reuse past experimental data, safety-constrained exploration to penalize experiments that cause excessive wear on equipment, or multi-agent coordination between parallel laboratories. On a more technical level, we also did not utilize curriculum-based scheduling to accelerate model learning, self-supervised (contrastive) model learning, or advanced probabilistic modeling techniques like flow matching and diffusion.

Despite the immense promise of **AI**, developing powerful systems is a multifaceted, labor-intensive, time-consuming, and economically expensive endeavor. Learning algorithms frequently exhibit instability or brittleness in practice, performing worse than suggested by *in-silico* benchmarks or experiments. Identifying shortcomings is crucial, but finding practical solutions is often slow. Thus, we should stay vigilant on what we prioritize: do we accept quick patches for short-term gain? Or do we want to genuinely fix underlying issues? Our push towards **RL** in self-driving laboratories perhaps subscribes to this latter ambition. That being said, although deep **RL** is impressive, its brittle downstream behavior, the need for specialized development expertise, and high upfront costs must be key factors in business decisions regarding its adoption. In other words, a balance must exist in the development of future methods that weighs the aforementioned consequences and challenges accordingly to the implementation of simpler short-term methods.

In conclusion, we expect the continued development of general **AI** will ultimately serve a common good. While the path ahead may be uncertain, we still encourage the field to remain ambitious and optimistic in pursuing the aim of bettering **AI**, both on a fundamental level and when applied in the real world.

Bibliography

Abdolmaleki, A., Springenberg, J. T., Tassa, Y., Munos, R., Heess, N., & Riedmiller, M. (2018). Maximum a Posteriori Policy Optimisation. *International Conference on Learning Representations*.

Abdulsamad, H., Iqbal, S., & Särkkä, S. (2025). Sequential Monte Carlo for Policy Optimization in Continuous POMDPs.

Abolhasani, M., & Kumacheva, E. (2023). The rise of self-driving labs in chemical and materials sciences. *Nature Synthesis*, *2*(6), 483–492. https://doi.org/10.1038/s44160-022-00231-0

Agrawal, A., & Domke, J. (2021). Amortized Variational Inference for Simple Hierarchical Models. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in Neural Information Processing Systems* (pp. 21388–21399, Vol. 34). Curran Associates, Inc.

Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete Problems in AI Safety. arXiv preprint arXiv:1606.06565.

Amos, B. (2023). Tutorial on Amortized Optimization [Publisher: Now Publishers, Inc.]. *Foundations and Trends® in Machine Learning*, *16*(5), 592–732. https://doi.org/10.1561/2200000102

Andrychowicz, M., Denil, M., Gómez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., & de Freitas, N. (2016). Learning to learn by gradient descent by gradient descent. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 29). Curran Associates, Inc.

Antonoglou, I., Schrittwieser, J., Ozair, S., Hubert, T. K., & Silver, D. (2022). Planning in Stochastic Environments with a Learned Model. *International Conference on Learning Representations*.

Asmussen, S., & Glynn, P. W. (2007). *Stochastic Simulation: Algorithms and Analysis* (1st, Vol. 57). Springer. https://doi.org/10.1007/978-0-387-69033-9

Åström, K. J. (Ed.). (1970). *Introduction to Stochastic Control Theory* (Vol. 70). Elsevier Science.

Attias, H. (2003). Planning by Probabilistic Inference. In C. M. Bishop & B. J. Frey (Eds.), *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics* (pp. 9–16, Vol. R4). PMLR.

Audi, R. (2010, September). *Epistemology: A Contemporary Introduction to the Theory of Knowledge* (3rd ed.). Routledge. https://doi.org/10.4324/9780203846469

Balestriero, R., Pesenti, J., & LeCun, Y. (2021). Learning in High Dimension Always Amounts to Extrapolation. arXiv preprint arXiv:2110.09485.

Barreto, A., Dabney, W., Munos, R., Hunt, J. J., Schaul, T., van Hasselt, H. P., & Silver, D. (2017). Successor Features for Transfer in Reinforcement Learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 30). Curran Associates, Inc.

Bauer, J., Baumli, K., Behbahani, F., Bhoopchand, A., Bradley-Schmieg, N., Chang, M., Clay, N., Collister, A., Dasagi, V., Gonzalez, L., Gregor, K., Hughes, E., Kashem, S., Loks-Thompson, M., Openshaw, H., Parker-Holder, J., Pathak, S., Perez-Nieves, N., Rakicevic, N., … Zhang, L. M. (2023). Human-Timescale Adaptation in an Open-Ended Task Space. *Proceedings of the 40th International Conference on Machine Learning*, 1887–1935.

Bechtle, S., Molchanov, A., Chebotar, Y., Grefenstette, E., Righetti, L., Sukhatme, G., & Meier, F. (2021). Meta Learning via Learned Loss. *International Conference on Pattern Recognition*, 4161–4168. https://doi.org/10.1109/ICPR48806.2021.9412010

Beck, J., Vuorio, R., Liu, E. Z., Xiong, Z., Zintgraf, L., Finn, C., & Whiteson, S. (2023). A Survey of Meta-Reinforcement Learning. arXiv preprint arXiv:2301.08028.

Becker, P., Freymuth, N., & Neumann, G. (2024). KalMamba: Towards Efficient Probabilistic State Space Models for RL under Uncertainty. *ICML 2024 Workshop: Aligning Reinforcement Learning Experimentalists and Theorists*.

Bellemare, M. G., Dabney, W., & Munos, R. (2017). A Distributional Perspective on Reinforcement Learning. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th international conference on machine learning* (pp. 449–458, Vol. 70). PMLR.

Bellman, R. E. (1957). *Dynamic Programming* (Rand Corporation, Ed.). Princeton University Press. https://doi.org/10.2307/j.ctv1nxcw0f

Bengio, Y., Louradour, J., Collobert, R., & Weston, J. (2009). Curriculum learning. *Proceedings of the 26th Annual International Conference on Machine Learning*, 41–48. https://doi.org/10.1145/1553374.1553380

Bergstra, J., & Bengio, Y. (2012). Random Search for Hyper-Parameter Optimization. *Journal of Machine Learning Research*, *13*(10), 281–305.

Bertsekas, D. (2019). *Reinforcement Learning and Optimal Control.* Athena Scientific.

Bertsekas, D. (2022). *Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control.* Athena Scientific.

Bishop, C. M. (2007). *Pattern Recognition and Machine Learning* (1st). Springer.

Blattmann, A., Rombach, R., Ling, H., Dockhorn, T., Kim, S. W., Fidler, S., & Kreis, K. (2023). Align your Latents: High-Resolution Video Synthesis with Latent Diffusion Models. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 22563–22575.

Bonnet, C., Luo, D., Byrne, D. J., Surana, S., Abramowitz, S., Duckworth, P., Coyette, V., Midgley, L. I., Tegegn, E., Kalloniatis, T., Mahjoub, O., Macfarlane, M., Smit, A. P., Grinsztajn, N., Boige, R., Waters, C. N., Mimouni, M. A. A., Sob, U. A. M., de Kock, R. J., … Laterre, A. (2024). Jumanji: A Diverse Suite of Scalable Reinforcement Learning Environments in JAX. *International Conference on Learning Representations*.

Botev, A., Ritter, H., & Barber, D. (2017, August). Practical Gauss-Newton Optimisation for Deep Learning. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th International Conference on Machine Learning* (pp. 557–565, Vol. 70). PMLR.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., & Zhang, Q. (2018). JAX: Composable transformations of Python+NumPy programs.

Bromiley, P. (2003). Products and Convolutions of Gaussian Probability Density Functions. *Tina-Vision Memo*, *3*(4), 1.

Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., & Colton, S. (2012). A Survey of Monte Carlo Tree Search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, *4*(1), 1–43. https://doi.org/10.1109/TCIAIG.2012.2186810

Carleo, G., Cirac, I., Cranmer, K., Daudet, L., Schuld, M., Tishby, N., Vogt-Maranto, L., & Zdeborová, L. (2019). Machine learning and the physical sciences. *Reviews of Modern Physics*, *91*(4), 045002. https://doi.org/10.1103/RevModPhys.91.045002

Casella, G., & Robert, C. P. (1996). Rao-Blackwellisation of Sampling Schemes. *Biometrika*, *83*(1), 81–94.

Chan, A., Silva, H., Lim, S., Kozuno, T., Mahmood, A. R., & White, M. (2022). Greedification Operators for Policy Optimization: Investigating Forward and Reverse KL Divergences. *Journal of Machine Learning Research*, *23*(253), 1–79.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., & Duvenaud, D. K. (2018). Neural Ordinary Differential Equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 31). Curran Associates, Inc.

Chen, Y., Hoffman, M. W., Colmenarejo, S. G., Denil, M., Lillicrap, T. P., Botvinick, M., & de Freitas, N. (2017, August). Learning to Learn without Gradient Descent by Gradient Descent. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th International Conference on Machine Learning* (pp. 748–756, Vol. 70). PMLR.

Chopin, N., & Papaspiliopoulos, O. (2020). *An Introduction to Sequential Monte Carlo* (1st). Springer. https://doi.org/10.1007/978-3-030-47845-2

Chow, Y., Nachum, O., Duenez-Guzman, E., & Ghavamzadeh, M. (2018). A Lyapunov-based Approach to Safe Reinforcement Learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 31). Curran Associates, Inc.

Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A. C., & Bengio, Y. (2015). A Recurrent Latent Variable Model for Sequential Data. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 28). Curran Associates, Inc.

Conn, A. R., Scheinberg, K., & Vicente, L. N. (2009). *Introduction to Derivative-Free Optimization*. Society for Industrial; Applied Mathematics. https://doi.org/10.1137/1.9780898718768

Cooper, G. F. (1990). The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, *42*(2), 393–405. https://doi.org/10.1016/0004-3702(90)90060-D

Cox, D. R. (2006). *Principles of Statistical Inference*. Cambridge University Press. https://doi.org/10.1017/CBO9780511813559

Cox, R. T. (1946). Probability, Frequency and Reasonable Expectation. *American Journal of Physics*, *14*(1), 1–13. https://doi.org/10.1119/1.1990764

Cremer, C., Li, X., & Duvenaud, D. (2018). Inference Suboptimality in Variational Autoencoders. *Proceedings of the 35th International Conference on Machine Learning*, 1078–1086.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, *2*(4), 303–314. https://doi.org/10.1007/BF02551274

Danihelka, I., Guez, A., Schrittwieser, J., & Silver, D. (2022). Policy improvement by planning with Gumbel. *International Conference on Learning Representations*.

Daulton, S., Balandat, M., & Bakshy, E. (2020). Differentiable Expected Hypervolume Improvement for Parallel Multi-Objective Bayesian Optimization. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems* (pp. 9851–9864, Vol. 33). Curran Associates, Inc.

Daxberger, E., Kristiadi, A., Immer, A., Eschenhagen, R., Bauer, M., & Hennig, P. (2021). Laplace Redux - Effortless Bayesian Deep Learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in Neural Information Processing Systems* (pp. 20089–20103, Vol. 34). Curran Associates, Inc.

de Vries, J. A., He, J., de Weerdt, M., & Spaan, M. T. J. (2025). Bayesian Meta-Reinforcement Learning with Laplace Variational Recurrent Networks. *Reinforcement Learning Conference*.

de Vries, J. A., He, J., Oren, Y., & Spaan, M. T. J. (2025). Trust-Region Twisted Policy Improvement. *Forty-second International Conference on Machine Learning*.

de Vries, J. A., He, J., Oren, Y., van der Vaart, P. R., de Weerdt, M. M., & Spaan, M. T. J. (2026). VariBASed: Variational Bayes-Adaptive Sequential Monte-Carlo Planning for Deep Reinforcement Learning. arXiv preprint arXiv:2602.18857.

de Vries, J. A., Voskuil, K., Moerland, T. M., & Plaat, A. (2021). Visualizing MuZero Models. *ICML 2021 Workshop on Unsupervised Reinforcement Learning*.

Deffayet, R., Thonet, T., Renders, J.-M., & de Rijke, M. (2023). Generative Slate Recommendation with Reinforcement Learning. *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, 580–588. https://doi.org/10.1145/3539597.3570412

Del Moral, P. (2004). *Feynman-Kac Formulae*. Springer. https://doi.org/10.1007/978-1-4684-9393-1

Dohare, S., Hernandez-Garcia, J. F., Lan, Q., Rahman, P., Mahmood, A. R., & Sutton, R. S. (2024). Loss of plasticity in deep continual learning. *Nature*, *632*(8026), 768–774. https://doi.org/10.1038/s41586-024-07711-7

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *International Conference on Learning Representations*.

Duan, Y., Schulman, J., Chen, X., Bartlett, P. L., Sutskever, I., & Abbeel, P. (2016). RL^2: Fast Reinforcement Learning via Slow Reinforcement Learning. arXiv preprint arXiv:1611.02779.

Duff, M. O. (2002). *Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes* [Doctoral dissertation, The University of Massachusetts Amherst]. ProQuest Dissertations & Theses Global.

Dulac-Arnold, G., Levine, N., Mankowitz, D. J., Li, J., Paduraru, C., Gowal, S., & Hester, T. (2021). Challenges of real-world reinforcement learning: Definitions, benchmarks and analysis. *Machine Learning*, *110*(9), 2419–2468. https://doi.org/10.1007/s10994-021-05961-4

Efron, B. (1987). Better Bootstrap Confidence Intervals. *Journal of the American Statistical Association*, *82*(397), 171–185. https://doi.org/10.2307/2289144

Eiben, A. E., & Smith, J. E. (2015). *Introduction to Evolutionary Computing* (2nd ed.). Springer-Verlag. https://doi.org/10.1007/978-3-662-44874-8

Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., Novikov, A., R. Ruiz, F. J., Schrittwieser, J., Swirszcz, G., Silver, D., Hassabis, D., & Kohli, P. (2022a). Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, *610*(7930), 47–53. https://doi.org/10.1038/s41586-022-05172-4

Fawzi, A., Balog, M., Huang, A., Hubert, T., Romera-Paredes, B., Barekatain, M., Novikov, A., R. Ruiz, F. J., Schrittwieser, J., Swirszcz, G., Silver, D., Hassabis, D., & Kohli, P. (2022b). Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, *610*(7930), 47–53. https://doi.org/10.1038/s41586-022-05172-4

Finn, C., Abbeel, P., & Levine, S. (2017, August). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In D. Precup & Y. W. Teh (Eds.), *Proceedings of the 34th International Conference on Machine Learning* (pp. 1126–1135, Vol. 70). PMLR.

Finn, C., Xu, K., & Levine, S. (2018). Probabilistic Model-Agnostic Meta-Learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 31). Curran Associates, Inc.

Finney, D. J. (1943). The Fractional Replication of Factorial Arrangements. *Annals of Eugenics*, *12*(1), 291–301. https://doi.org/10.1111/j.1469-1809.1943.tb02333.x

Fisher, R. A. (1935). *The Design of Experiments*. Oliver & Boyd.

Frankle, J., & Carbin, M. (2019). The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. *International Conference on Learning Representations*.

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., & Bachem, O. (2021). Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation. *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S. M. A., & Teh, Y. W. (2018). Neural Processes. arXiv preprint arXiv:1807.01622.

Geist, M., Scherrer, B., & Pietquin, O. (2019, June). A Theory of Regularized Markov Decision Processes. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th International Conference on Machine Learning* (pp. 2160–2169, Vol. 97). PMLR.

Gershman, S., & Goodman, N. (2014). Amortized Inference in Probabilistic Reasoning. *Proceedings of the Annual Meeting of the Cognitive Science Society*, *36*.

Gevers, M. (2005). Identification for Control: From the Early Achievements to the Revival of Experiment Design. *European Journal of Control*, *11*(4), 335–352. https://doi.org/10.3166/ejc.11.335-352

Ghavamzadeh, M., Mannor, S., Pineau, J., & Tamar, A. (2015). Bayesian Reinforcement Learning: A Survey. *Foundations and Trends® in Machine Learning*, *8*(5-6), 359–483. https://doi.org/10.1561/2200000049

González-Duque, M., Michael, R., Bartels, S., Zainchkovskyy, Y., Hauberg, S., & Boomsma, W. (2024a). A survey and benchmark of high-dimensional Bayesian optimization of discrete sequences. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, & C. Zhang (Eds.), *Advances in neural information processing systems* (pp. 140478–140508, Vol. 37). Curran Associates, Inc.

González-Duque, M., Michael, R., Bartels, S., Zainchkovskyy, Y., Hauberg, S., & Boomsma, W. (2024b). A survey and benchmark of high-dimensional Bayesian optimization of discrete sequences. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, & C. Zhang (Eds.), *Advances in neural information processing systems* (pp. 140478–140508, Vol. 37). Curran Associates, Inc. https://doi.org/10.52202/079017-4459

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning* (F. Bach, Ed.). The MIT Press.

Gordon, J., Bronskill, J., Bauer, M., Nowozin, S., & Turner, R. (2019). Meta-Learning Probabilistic Inference for Prediction. *International Conference on Learning Representations*.

Goyal, A., Sordoni, A., Côté, M.-A., Ke, N. R., & Bengio, Y. (2017). Z-Forcing: Training Stochastic Recurrent Networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 30). Curran Associates, Inc.

Grant, E., Finn, C., Levine, S., Darrell, T., & Griffiths, T. (2018). Recasting Gradient-Based Meta-Learning as Hierarchical Bayes. *International Conference on Learning Representations*.

Greenberg, I., Mannor, S., Chechik, G., & Meirom, E. (2023). Train Hard, Fight Easy: Robust Meta Reinforcement Learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, & S. Levine (Eds.), *Advances in Neural Information Processing Systems* (pp. 68276–68299, Vol. 36). Curran Associates, Inc.

Gregor, K., Jimenez Rezende, D., Besse, F., Wu, Y., Merzic, H., & van den Oord, A. (2019). Shaping Belief States with Generative Environment Models for RL. *Advances in Neural Information Processing Systems*, *32*.

Grill, J.-B., Altché, F., Tang, Y., Hubert, T., Valko, M., Antonoglou, I., & Munos, R. (2020). Monte-Carlo Tree Search as Regularized Policy Optimization. *Proceedings of the 37th International Conference on Machine Learning*, *119*, 3769–3778.

Gu, A., & Dao, T. (2024). Mamba: Linear-Time Sequence Modeling with Selective State Spaces. *First Conference on Language Modeling*.

Gu, S., Ghahramani, Z., & Turner, R. E. (2015). Neural Adaptive Sequential Monte Carlo. *Advances in Neural Information Processing Systems*, *28*.

Guez, A., Mirza, M., Gregor, K., Kabra, R., Racaniere, S., Weber, T., Raposo, D., Santoro, A., Orseau, L., Eccles, T., Wayne, G., Silver, D., & Lillicrap, T. (2019, June). An Investigation of Model-Free Planning. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th International Conference on Machine Learning* (pp. 2464–2473, Vol. 97). PMLR.

Gupta, A., Mendonca, R., Liu, Y., Abbeel, P., & Levine, S. (2018). Meta-Reinforcement Learning of Structured Exploration Strategies. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 31). Curran Associates, Inc.

Haarnoja, T., Zhou, A., Abbeel, P., & Levine, S. (2018). Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *Proceedings of the 35th International Conference on Machine Learning*, *80*, 1861–1870.

Hafner, D., Lillicrap, T., Ba, J., & Norouzi, M. (2020). Dream to Control: Learning Behaviors by Latent Imagination. *International Conference on Learning Representations*.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., & Davidson, J. (2019). Learning Latent Dynamics for Planning from Pixels. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th International Conference on Machine Learning* (pp. 2555–2565, Vol. 97). PMLR.

Halevy, A., Norvig, P., & Pereira, F. (2009). The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems*. https://doi.org/10.1109/MIS.2009.36

Hallak, A., Castro, D. D., & Mannor, S. (2015). Contextual Markov Decision Processes. arXiv preprint arXiv:1502.02259.

Hamrick, J. B., Friesen, A. L., Behbahani, F., Guez, A., Viola, F., Witherspoon, S., Anthony, T., Buesing, L. H., Veličković, P., & Weber, T. (2021). On the role of planning in model-based deep reinforcement learning. *International Conference on Learning Representations*.

Hansen, N. A., Su, H., & Wang, X. (2022, July). Temporal Difference Learning for Model Predictive Control. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, & S. Sabato (Eds.), *Proceedings of the 39th International Conference on Machine Learning* (pp. 8387–8406, Vol. 162). PMLR.

Hansen, N. A., Su, H., & Wang, X. (2024). TD-MPC2: Scalable, Robust World Models for Continuous Control. *International Conference on Learning Representations*.

Häse, F., Roch, L., & Aspuru-Guzik, A. (2019). Next-Generation Experimentation with Self-Driving Laboratories. *Trends in Chemistry*, *1*(3), 282–291. https://doi.org/10.1016/j.trechm.2019.02.007

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer. https://doi.org/10.1007/978-0-387-84858-7

He, J., Moerland, T. M., de Vries, J. A., & Oliehoek, F. A. (2024). What model does MuZero learn? In U. Endriss, F. S. Melo, K. Bach, A. J. B. Diz, J. M. Alonso-Moral, S. Barro, & F. Heintz (Eds.), *ECAI 2024 - 27th European Conference on Artificial Intelligence,* (pp. 1599–1606, Vol. 392). IOS Press. https://doi.org/10.3233/FAIA240666

Heek, J., Levskaya, A., Oliver, A., Ritter, M., Rondepierre, B., Steiner, A., & van Zee, M. (2024). Flax: A neural network library and ecosystem for JAX.

Hennig, P., & Schuler, C. J. (2012). Entropy Search for Information-Efficient Global Optimization. *Journal of Machine Learning Research*, *13*(57), 1809–1837.

Ho, Y.-C., Zhao, Q.-C., & Jia, Q.-S. (2007). *Ordinal Optimization*. Springer US. https://doi.org/10.1007/978-0-387-68692-9

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hoffman, M., Doucet, A., Freitas, N., & Jasra, A. (2007). Bayesian Policy Learning with Trans-Dimensional MCMC. In J. Platt, D. Koller, Y. Singer, & S. Roweis (Eds.), *Advances in Neural Information Processing Systems* (Vol. 20). Curran Associates, Inc.

Hornby, G., Globus, A., Linden, D., & Lohn, J. (2006). Automated Antenna Design with Evolutionary Algorithms. In *Space 2006*. American Institute of Aeronautics; Astronautics. https://doi.org/10.2514/6.2006-7242

Hospedales, T. M., Antoniou, A., Micaelli, P., & Storkey, A. J. (2021). Meta-Learning in Neural Networks: A Survey. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, *44*(9), 5149–5169. https://doi.org/10.1109/TPAMI.2021.3079209

Hubert, T., Schrittwieser, J., Antonoglou, I., Barekatain, M., Schmitt, S., & Silver, D. (2021). Learning and Planning in Complex Action Spaces. *Proceedings of the 38th International Conference on Machine Learning*, *139*, 4476–4486.

Hutter, F., Kotthoff, L., & Vanschoren, J. (Eds.). (2019). *Automated Machine Learning: Methods, Systems, Challenges*. Springer International Publishing. https://doi.org/10.1007/978-3-030-05318-5

Igl, M., Zintgraf, L., Le, T. A., Wood, F., & Whiteson, S. (2018). Deep Variational Reinforcement Learning for POMDPs. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th International Conference on Machine Learning* (pp. 2117–2126, Vol. 80). PMLR.

Ilten, P., Williams, M., & Yang, Y. (2017). Event generator tuning using Bayesian optimization. *Journal of Instrumentation*, *12*(04), P04028. https://doi.org/10.1088/1748-0221/12/04/P04028

Immer, A., Korzepa, M., & Bauer, M. (2021, April). Improving predictions of Bayesian neural nets via local linearization. In A. Banerjee & K. Fukumizu (Eds.), *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics* (pp. 703–711, Vol. 130). PMLR.

Iqbal, S., Corenflos, A., Särkkä, S., & Abdulsamad, H. (2024). Nesting Particle Filters for Experimental Design in Dynamical Systems.

Jacot, A., Gabriel, F., & Hongler, C. (2018). Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 31). Curran Associates, Inc.

Jaderberg, M., Czarnecki, W. M., Dunning, I., Marris, L., Lever, G., Castañeda, A. G., Beattie, C., Rabinowitz, N. C., Morcos, A. S., Ruderman, A., Sonnerat, N., Green, T., Deason, L., Leibo, J. Z., Silver, D., Hassabis, D., Kavukcuoglu, K., & Graepel, T. (2019). Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science*, *364*(6443), 859–865. https://doi.org/10.1126/science.aau6249

Jain, A., Ong, S. P., Hautier, G., Chen, W., Richards, W. D., Dacek, S., Cholia, S., Gunter, D., Skinner, D., Ceder, G., & Persson, K. A. (2013). Commentary: The Materials Project: A materials genome approach to accelerating materials innovation. *APL Materials*, *1*(1), 011002. https://doi.org/10.1063/1.4812323

Jamieson, K., & Talwalkar, A. (2016). Non-stochastic Best Arm Identification and Hyperparameter Optimization. *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, 240–248.

Jaynes, E. T. (2003). *Probability Theory: The Logic of Science* (G. L. Bretthorst, Ed.). Cambridge University Press. https://doi.org/10.1017/CBO9780511790423

Jeong, J., Wang, X., Wang, J., Sanner, S., & Poupart, P. (2025). Reflect-then-Plan: Offline Model-Based Planning through a Doubly Bayesian Lens. *Forty-second International Conference on Machine Learning*.

Johnson, D. S. (1990, January). A Catalog of Complexity Classes. In J. Van leeuwen (Ed.), *Algorithms and Complexity* (pp. 67–161). Elsevier. https://doi.org/10.1016/B978-0-444-88071-0.50007-2

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., … Hassabis, D. (2021). Highly accurate protein structure prediction with AlphaFold. *Nature*, *596*(7873), 583–589. https://doi.org/10.1038/s41586-021-03819-2

Kajita, S., Kinjo, T., & Nishi, T. (2020). Autonomous molecular design by Monte-Carlo tree search and rapid evaluations using molecular dynamics simulations. *Communications Physics*, *3*(1), 1–11. https://doi.org/10.1038/s42005-020-0338-y

Kakade, S. M. (2003). *On the Sample Complexity of Reinforcement Learning* [Doctoral dissertation, UCL (University College London)]. ProQuest Dissertations & Theses Global.

Kalman, R. E. (1960). A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, *82*(1), 35–45. https://doi.org/10.1115/1.3662552

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). Scaling Laws for Neural Language Models. arXiv preprint arXiv:2001.08361.

Katharopoulos, A., Vyas, A., Pappas, N., & Fleuret, F. (2020). Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. *Proceedings of the 37th International Conference on Machine Learning*, 5156–5165.

Kaufmann, E., Bauersfeld, L., Loquercio, A., Müller, M., Koltun, V., & Scaramuzza, D. (2023). Champion-level drone racing using deep reinforcement learning. *Nature*, *620*(7976), 982–987. https://doi.org/10.1038/s41586-023-06419-4

Khetarpal, K., Riemer, M., Rish, I., & Precup, D. (2022). Towards Continual Reinforcement Learning: A Review and Perspectives. *Journal of Artificial Intelligence Research*, *75*, 1401–1476. https://doi.org/10.1613/jair.1.13673

Kiefer, J. (1959). Optimum Experimental Designs. *Journal of the Royal Statistical Society: Series B (Methodological)*, *21*(2), 272–304. https://doi.org/10.1111/j.2517-6161.1959.tb00338.x

Kim, Y., Wiseman, S., Miller, A., Sontag, D., & Rush, A. (2018). Semi-Amortized Variational Autoencoders. *Proceedings of the 35th International Conference on Machine Learning*, 2678–2687.

Kingma, D. P., & Ba, J. (2017). Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*.

Kingma, D. P., & Welling, M. (2014). Auto-Encoding Variational Bayes. *International Conference on Learning Representations*.

Kirk, R., Zhang, A., Grefenstette, E., & Rocktäschel, T. (2023). A Survey of Zero-shot Generalisation in Deep Reinforcement Learning. *Journal of Artificial Intelligence Research*, *76*, 201–264. https://doi.org/10.1613/jair.1.14174

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., & Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, *114*(13), 3521–3526. https://doi.org/10.1073/pnas.1611835114

Lacoste–Julien, S., Huszár, F., & Ghahramani, Z. (2011). Approximate inference for the loss-calibrated Bayesian. In G. Gordon, D. Dunson, & M. Dudík (Eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics* (pp. 416–424, Vol. 15). PMLR.

Langosco, L. L. D., Koch, J., Sharkey, L. D., Pfau, J., & Krueger, D. (2022). Goal Misgeneralization in Deep Reinforcement Learning. *Proceedings of the 39th International Conference on Machine Learning*, 12004–12019.

Lawson, D., Tucker, G., Naesseth, C. A., Maddison, C., Adams, R. P., & Teh, Y. W. (2018). Twisted Variational Sequential Monte Carlo. *Third workshop on Bayesian Deep Learning (NeurIPS)*.

Lehmann, E., & Casella, G. (1998). *Theory of Point Estimation* (2nd). Springer-Verlag. https://doi.org/10.1007/b98854

Levine, S. (2018). Reinforcement Learning and Control as Probabilistic Inference: Tutorial and Review. arXiv preprint arXiv:1805.00909.

Lew, A. K., Zhi-Xuan, T., Grand, G., & Mansinghka, V. K. (2023). Sequential Monte Carlo Steering of Large Language Models using Probabilistic Programs.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., & Talwalkar, A. (2018). Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization. *Journal of Machine Learning Research*, *18*(185), 1–52.

Lioutas, V., Lavington, J. W., Sefas, J., Niedoba, M., Liu, Y., Zwartsenberg, B., Dabiri, S., Wood, F., & Scibior, A. (2023). Critic Sequential Monte Carlo. *International Conference on Learning Representations*.

Liu, Q., Li, L., Tang, Z., & Zhou, D. (2018). Breaking the Curse of Horizon: Infinite-Horizon Off-Policy Estimation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 31). Curran Associates, Inc.

Loshchilov, I., & Hutter, F. (2019). Decoupled Weight Decay Regularization. *International Conference on Learning Representations*.

Lu, C., Kuba, J., Letcher, A., Metz, L., Schroeder de Witt, C., & Foerster, J. (2022). Discovered Policy Optimisation. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, & A. Oh (Eds.), *Advances in Neural Information Processing Systems* (pp. 16455–16468, Vol. 35). Curran Associates, Inc.

Lu, C., Lu, C., Lange, R. T., Foerster, J., Clune, J., & Ha, D. (2024). The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery. arXiv preprint arXiv:2408.06292.

Lu, C., Schroecker, Y., Gu, A., Parisotto, E., Foerster, J., Singh, S., & Behbahani, F. (2023). Structured State Space Models for In-Context Reinforcement Learning. *Advances in Neural Information Processing Systems*, *36*.

Luxburg, U. v., & Schölkopf, B. (2011, January). Statistical Learning Theory: Models, Concepts, and Results. In D. M. Gabbay, S. Hartmann, & J. Woods (Eds.), *Handbook of the History of Logic* (pp. 651–706, Vol. 10). North-Holland. https://doi.org/10.1016/B978-0-444-52936-7.50016-1

Macfarlane, M., Toledo, E., Byrne, D. J., Duckworth, P., & Laterre, A. (2024). SPO: Sequential Monte Carlo Policy Optimisation. *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

MacKay, D. J. C. (1992). Bayesian Interpolation. *Neural Computation*, *4*(3), 415–447. https://doi.org/10.1162/neco.1992.4.3.415

Malach, E., Yehudai, G., Shalev-Schwartz, S., & Shamir, O. (2020). Proving the Lottery Ticket Hypothesis: Pruning is All You Need. *Proceedings of the 37th International Conference on Machine Learning*, 6682–6691.

Mankowitz, D. J., Michi, A., Zhernov, A., Gelmi, M., Selvi, M., Paduraru, C., Leurent, E., Iqbal, S., Lespiau, J.-B., Ahern, A., Köppe, T., Millikin, K., Gaffney, S., Elster, S., Broshear, J., Gamble, C., Milan, K., Tung, R., Hwang, M., … Silver, D. (2023a). Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, *618*(7964), 257–263. https://doi.org/10.1038/s41586-023-06004-9

Mankowitz, D. J., Michi, A., Zhernov, A., Gelmi, M., Selvi, M., Paduraru, C., Leurent, E., Iqbal, S., Lespiau, J.-B., Ahern, A., Köppe, T., Millikin, K., Gaffney, S., Elster, S., Broshear, J., Gamble, C., Milan, K., Tung, R., Hwang, M., … Silver, D. (2023b). Faster sorting algorithms discovered using deep reinforcement learning. *Nature*, *618*(7964), 257–263. https://doi.org/10.1038/s41586-023-06004-9

Margossian, C., & Blei, D. (2024). Amortized Variational Inference: When and Why? *The 40th Conference on Uncertainty in Artificial Intelligence*.

Martens, J., & Grosse, R. (2015, July). Optimizing Neural Networks with Kronecker-factored Approximate Curvature. In F. Bach & D. Blei (Eds.), *Proceedings of the 32nd International Conference on Machine Learning* (pp. 2408–2417, Vol. 37). PMLR.

Merchant, A., Batzner, S., Schoenholz, S. S., Aykol, M., Cheon, G., & Cubuk, E. D. (2023). Scaling deep learning for materials discovery. *Nature*, *624*(7990), 80–85. https://doi.org/10.1038/s41586-023-06735-9

Mikulik, V., Delétang, G., McGrath, T., Genewein, T., Martic, M., Legg, S., & Ortega, P. (2020). Meta-trained agents implement Bayes-optimal agents. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, & H. Lin (Eds.), *Advances in Neural Information Processing Systems* (pp. 18691–18703, Vol. 33). Curran Associates, Inc.

Mnih, A., & Gregor, K. (2014). Neural Variational Inference and Learning in Belief Networks. *Proceedings of the 31st International Conference on Machine Learning*, 1791–1799.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. https://doi.org/10.1038/nature14236

Mockus, J. (1989). *Bayesian Approach to Global Optimization: Theory and Applications* (M. Hazewinkel, Ed.; Vol. 37). Springer Netherlands. https://doi.org/10.1007/978-94-009-0909-0

Moral, P. D., Doucet, A., & Singh, S. (2010). Forward Smoothing using Sequential Monte Carlo. arXiv preprint arXiv:1012.5390.

Morimoto, J., & Doya, K. (2005). Robust Reinforcement Learning. *Neural Computation*, *17*(2), 335–359. https://doi.org/10.1162/0899766053011528

Munos, R. (2014). From Bandits to Monte-Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning. *Foundations and Trends® in Machine Learning*, *7*(1), 1–129. https://doi.org/10.1561/2200000038

Munos, R., Stepleton, T., Harutyunyan, A., & Bellemare, M. (2016). Safe and Efficient Off-Policy Reinforcement Learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 29). Curran Associates, Inc.

Naesseth, C. A., Linderman, S., Ranganath, R., & Blei, D. (2018). Variational Sequential Monte Carlo. *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, *84*, 968–977.

Naesseth, C. A., Lindsten, F., & Schön, T. B. (2019). Elements of Sequential Monte Carlo. *Foundations and Trends® in Machine Learning*, *12*(3), 307–392. https://doi.org/10.1561/2200000074

Neal, R. M., & Hinton, G. E. (1998). A View of the EM Algorithm that Justifies Incremental, Sparse, and other Variants. In *Learning in Graphical Models* (pp. 355–368). Springer Netherlands. https://doi.org/10.1007/978-94-011-5014-9_12

Nelles, O. (2001). *Nonlinear System Identification*. Springer. https://doi.org/10.1007/978-3-662-04323-3

Nocedal, J., & Wright, S. J. (2006). *Numerical Optimization* (2nd ed.). Springer New York. https://doi.org/10.1007/978-0-387-40065-5

Okasha, S. (2016, July). *Philosophy of Science: Very Short Introduction*. Oxford University Press. https://doi.org/10.1093/actrade/9780198745587.001.0001

Øksendal, B. (2003). *Stochastic Differential Equations*. Springer. https://doi.org/10.1007/978-3-642-14394-6

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., … Zoph, B. (2024). GPT-4 Technical Report. arXiv preprint arXiv:2303.08774.

Oren, Y., de Vries, J. A., van der Vaart, P. R., Spaan, M. T. J., & Böhmer, W. (2025). Parallelizing Tree Search with Twice Sequential Monte Carlo. arXiv preprint arXiv:2511.14220.

Oren, Y., Vadocz, V., Spaan, M. T. J., & Boehmer, W. (2025). Epistemic Monte Carlo Tree Search. *The Thirteenth International Conference on Learning Representations*.

Osband, I., & Roy, B. V. (2017). Why is Posterior Sampling Better than Optimism for Reinforcement Learning? *Proceedings of the 34th International Conference on Machine Learning*, 2701–2710.

Osband, I., Russo, D., & Van Roy, B. (2013). (More) Efficient Reinforcement Learning via Posterior Sampling. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Weinberger (Eds.), *Advances in Neural Information Processing Systems* (Vol. 26). Curran Associates, Inc.

Pan, A., Bhatia, K., & Steinhardt, J. (2021). The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models. *Deep RL Workshop NeurIPS 2021*.

Papadimitriou, C. H., & Tsitsiklis, J. N. (1987). The Complexity of Markov Decision Processes. *Mathematics of Operations Research*, *12*(3), 441–450.

Parisotto, E., & Salakhutdinov, R. (2021). Efficient Transformers in Reinforcement Learning using Actor-Learner Distillation. *International Conference on Learning Representations*.

Patterson, A., Neumann, S., White, M., & White, A. (2024). Empirical Design in Reinforcement Learning. *Journal of Machine Learning Research*, *25*(318), 1–63.

Pearl, J. (1988, August). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc.

Peirce, C. S. (1878). How to Make Our Ideas Clear. *Popular Science Monthly*, *12*(Jan.), 286–302.

Peirce, C. S. (1876). Note on the Theory of the Economy of Research. *United States Coast Survey Report*. https://doi.org/10.1287/opre.15.4.643

Peirce, C. S. (1883). A theory of probable inference. In *Studies in logic by members of the Johns Hopkins University*. Little, Brown; Co. https://doi.org/10.1037/12811-007

Piché, A., Thomas, V., Ibrahim, C., Bengio, Y., & Pal, C. (2019). Probabilistic Planning with Sequential Monte Carlo methods. *International Conference on Learning Representations*.

Pitt, M. K., & Shephard, N. (1999). Filtering via Simulation: Auxiliary Particle Filters. *Journal of the American Statistical Association*, *94*(446), 590–599. https://doi.org/10.1080/01621459.1999.10474153

Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Heek, J., Xiao, K., Agrawal, S., & Dean, J. (2023). Efficiently Scaling Transformer Inference. In D. Song, M. Carbin, & T. Chen (Eds.), *Proceedings of machine learning and systems* (pp. 606–624, Vol. 5). Curan.

Powell, W. B. (2021). From Reinforcement Learning to Optimal Control: A Unified Framework for Sequential Decisions. In K. G. Vamvoudakis, Y. Wan, F. L. Lewis, & D. Cansever (Eds.), *Handbook of Reinforcement Learning and Control* (pp. 29–74). Springer International Publishing. https://doi.org/10.1007/978-3-030-60990-0_3

Puterman, M. L. (1994, April). *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (1st). John Wiley & Sons, Inc. https://doi.org/10.1002/9780470316887

Radhakrishnan, A., Beaglehole, D., Pandit, P., & Belkin, M. (2024). Mechanism for feature learning in neural networks and backpropagation-free machine learning models. *Science*, *383*(6690), 1461–1467. https://doi.org/10.1126/science.adi5639

Rakelly, K., Zhou, A., Finn, C., Levine, S., & Quillen, D. (2019). Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables. *Proceedings of the 36th International Conference on Machine Learning*, *97*, 5331–5340.

Rasmussen, C. E., & Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning* (Vol. 2). The MIT Press. https://doi.org/10.7551/mitpress/3206.001.0001

Rawlik, K., Toussaint, M., & Vijayakumar, S. (2012). On Stochastic Optimal Control and Reinforcement Learning by Approximate Inference. In N. Roy, P. Newman, & S. Srinivasa (Eds.), *Robotics: Science and systems VIII*. MIT Press. https://doi.org/10.15607/RSS.2012.VIII.045

Ritter, H., Botev, A., & Barber, D. (2018). Online Structured Laplace Approximations for Overcoming Catastrophic Forgetting. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 31). Curran Associates, Inc.

Roch, L. M., Häse, F., Kreisbeck, C., Tamayo-Mendoza, T., Yunker, L. P. E., Hein, J. E., & Aspuru-Guzik, A. (2020). ChemOS: An orchestration software to democratize autonomous discovery. *PLOS ONE*, *15*(4), e0229862. https://doi.org/10.1371/journal.pone.0229862

Roijers, D. M., Vamplew, P., Whiteson, S., & Dazeley, R. (2013). A Survey of Multi-Objective Sequential Decision-Making. *Journal of Artificial Intelligence Research*, *48*, 67–113. https://doi.org/10.1613/jair.3987

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., & Ommer, B. (2022). High-Resolution Image Synthesis with Latent Diffusion Models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10674–10685. https://doi.org/10.1109/CVPR52688.2022.01042

Rosin, C. D. (2011). Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, *61*(3), 203–230. https://doi.org/10.1007/s10472-011-9258-6

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., & Silver, D. (2020). Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, *588*(7839), 604–609. https://doi.org/10.1038/s41586-020-03051-4

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015, July). Trust region policy optimization. In F. Bach & D. Blei (Eds.), *Proceedings of the 32nd International Conference on Machine Learning* (pp. 1889–1897, Vol. 37). PMLR.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347.

Ścibior, A., & Wood, F. (2021). Differentiable Particle Filtering without Modifying the Forward Pass.

Segler, M. H. S., Preuss, M., & Waller, M. P. (2018). Planning chemical syntheses with deep neural networks and symbolic AI. *Nature*, *555*(7698), 604–610. https://doi.org/10.1038/nature25978

Seijen, H., & Sutton, R. (2014). True Online TD(lambda). *Proceedings of the 31st International Conference on Machine Learning*, *32*, 692–700.

Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., & Pathak, D. (2020, July). Planning to Explore via Self-Supervised World Models. In H. D. III & A. Singh (Eds.), *Proceedings of the 37th International Conference on Machine Learning* (pp. 8583–8592, Vol. 119). PMLR.

Shi, F., & Evans, J. (2023). Surprising combinations of research contents and contexts are related to impact and emerge with scientific outsiders from distant disciplines. *Nature Communications*, *14*(1), 1641. https://doi.org/10.1038/s41467-023-36741-4

Shiryaev, A. N. (1978). *Optimal Stopping Rules* (B. Rozovskii & G. Grimmett, Eds.; Vol. 8). Springer. https://doi.org/10.1007/978-3-540-74011-7

Shumailov, I., Shumaylov, Z., Zhao, Y., Gal, Y., Papernot, N., & Anderson, R. (2023). The Curse of Recursion: Training on Generated Data Makes Models Forget. arXiv preprint arXiv:2305.17493.

Sikchi, H., Tirinzoni, A., Touati, A., Xu, Y., Kanervisto, A., Niekum, S., Zhang, A., Lazaric, A., & Pirotta, M. (2025). Fast Adaptation with Behavioral Foundation Models. *Reinforcement Learning Journal*.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489. https://doi.org/10.1038/nature16961

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., & Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, *362*(6419), 1140–1144. https://doi.org/10.1126/science.aar6404

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Driessche, G. v. d., Graepel, T., & Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, *550*(7676), 354. https://doi.org/10.1038/nature24270

Silver, D., & Sutton, R. S. (2025). Welcome to the Era of Experience.

Singh, G., Yoon, J., Son, Y., & Ahn, S. (2019). Sequential Neural Processes. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 32). Curran Associates, Inc.

Smith, J. T., Warrington, A., & Linderman, S. (2023). Simplified State Space Layers for Sequence Modeling. *The Eleventh International Conference on Learning Representations*.

Smith, K. (1918). On the Standard Deviations of Adjusted and Interpolated Values of an Observed Polynomial Function and its Constants and the Guidance they give Towards a Proper Choice of the Distribution of Observations. *Biometrika*. https://doi.org/10.2307/2331929

Spaan, M. T. J. (2012). Partially Observable Markov Decision Processes. In M. Wiering & M. van Otterlo (Eds.), *Reinforcement Learning: State-of-the-Art* (pp. 387–414). Springer. https://doi.org/10.1007/978-3-642-27645-3_12

Stanley, K. O., & Miikkulainen, R. (2002). Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation*, *10*(2), 99–127. https://doi.org/10.1162/106365602320169811

Strogatz, S. H. (2024, January). *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering* (3rd ed.). Chapman; Hall/CRC. https://doi.org/10.1201/9780429398490

Stuhlmülller, A., Hawkins, R. X. D., Siddharth, N., & Goodman, N. D. (2015). Coarse-to-Fine Sequential Monte Carlo for Probabilistic Programs. arXiv preprint arXiv:1509.02962.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd). A Bradford Book.

Svensson, A., Schön, T. B., & Kok, M. (2015). Nonlinear State Space Smoothing Using the Conditional Particle Filter. *IFAC-PapersOnLine*, *48*(28), 975–980. https://doi.org/10.1016/j.ifacol.2015.12.257

Theodorou, E., Buchli, J., & Schaal, S. (2010). A Generalized Path Integral Control Approach to Reinforcement Learning. *Journal of Machine Learning Research*, *11*(104), 3137–3181.

Touati, A., & Ollivier, Y. (2021). Learning One Representation to Optimize All Rewards. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in neural information processing systems* (pp. 13–23, Vol. 34). Curran Associates, Inc.

Toussaint, M. (2009). Robot trajectory optimization using approximate inference. *Proceedings of the 26th Annual International Conference on Machine Learning*, 1049–1056. https://doi.org/10.1145/1553374.1553508

Toussaint, M., Charlin, L., & Poupart, P. (2008). Hierarchical POMDP controller optimization by likelihood maximization. In D. A. McAllester & P. Myllymäki (Eds.), *Proceedings of the 24th conference on uncertainty in artificial intelligence* (pp. 562–570, Vol. R6). PMLR.

Toussaint, M., & Storkey, A. J. (2006). Probabilistic Inference for Solving Discrete and Continuous State Markov Decision Processes. *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 945–952.

van Hasselt, H., Madjiheurem, S., Hessel, M., Silver, D., Barreto, A., & Borsa, D. (2021). Expected Eligibility Traces. *Proceedings of the AAAI Conference on Artificial Intelligence*, *35*(11), 9997–10005. https://doi.org/10.1609/aaai.v35i11.17200

van Leeuwen, S. A. J. (2023). *Language Assistance in Reinforcement Learning in Dynamic Environments* [Master's thesis, Delft University of Technology]. TU Delft Repository.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 30). Curran Associates, Inc.

Wald, A. (1945). Sequential Tests of Statistical Hypotheses. *The Annals of Mathematical Statistics*, *16*(2), 117–186. https://doi.org/10.1214/aoms/1177731118

Waltz, D., & Buchanan, B. G. (2009). Automating Science. *Science*, *324*(5923), 43–44. https://doi.org/10.1126/science.1172781

Wang, C., Wu, Y., Vuong, Q., & Ross, K. (2020). Striving for Simplicity and Performance in Off-Policy DRL: Output Normalization and Non-Uniform Sampling. In H. D. III & A. Singh (Eds.), *Proceedings of the 37th International Conference on Machine Learning* (pp. 10070–10080, Vol. 119). PMLR.

Wang, J. X., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J. Z., Munos, R., Blundell, C., Kumaran, D., & Botvinick, M. (2017). Learning to reinforcement learn. arXiv preprint arXiv:1611.05763.

Wang, S., Liu, S., Ye, W., You, J., & Gao, Y. (2024). EfficientZero V2: Mastering Discrete and Continuous Control with Limited Data. arXiv preprint arXiv:2403.00564.

Williams, G., Aldrich, A., & Theodorou, E. (2015). Model Predictive Path Integral Control using Covariance Variable Importance Sampling. arXiv preprint arXiv:1509.01149.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, *8*(3-4), 229–256. https://doi.org/10.1007/BF00992696

Wu, M., Choi, K., Goodman, N., & Ermon, S. (2020). Meta-Amortized Variational Inference and Learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(4), 6404–6412. https://doi.org/10.1609/aaai.v34i04.6111

Xiong, Z., Zintgraf, L. M., Beck, J. A., Vuorio, R., & Whiteson, S. (2021). On the Practical Consistency of Meta-Reinforcement Learning Algorithms. *Fifth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems*.

Yang, J., Zhou, K., Li, Y., & Liu, Z. (2024). Generalized Out-of-Distribution Detection: A Survey. *International Journal of Computer Vision*, *132*(12), 5635–5662. https://doi.org/10.1007/s11263-024-02117-4

Ye, W., Liu, S., Kurutach, T., Abbeel, P., & Gao, Y. (2021). Mastering Atari Games with Limited Data. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, & J. W. Vaughan (Eds.), *Advances in Neural Information Processing Systems* (pp. 25476–25488, Vol. 34). Curran Associates, Inc.

Yildiz, C., Heinonen, M., & Lähdesmäki, H. (2021). Continuous-time model-based reinforcement learning. In M. Meila & T. Zhang (Eds.), *Proceedings of the 38th international conference on machine learning* (pp. 12009–12018, Vol. 139). PMLR.

Ziebart, B. D. (2010, July). *Modeling Purposeful Adaptive Behavior with the Principle of Maximum Causal Entropy* [Doctoral dissertation, Carnegie Mellon University]. https://doi.org/10.1184/R1/6720692.v1

Ziebart, B. D., Bagnell, J. A., & Dey, A. K. (2010). Modeling Interaction via the Principle of Maximum Causal Entropy. *Proceedings of the 27th International Conference on Machine Learning*, 1255–1262.

Zintgraf, L., Schulze, S., Lu, C., Feng, L., Igl, M., Shiarlis, K., Gal, Y., Hofmann, K., & Whiteson, S. (2021). VariBAD: Variational Bayes-Adaptive Deep RL via Meta-Learning. *Journal of Machine Learning Research*, *22*(289), 1–39.

Dankwoord

Het gevoel dat ik het ene moment relaxt en zonder duidelijke toekomstvisie door mijn schooljaren heen surf, en het volgende moment opeens een dankwoord schrijf voor een doctoraal proefschrift, is moeilijk in woorden te vatten. Je begint aan een groot doel als het behalen van een PhD met een gevoel van prestatiedrang, maar door het lange traject begint elk niveau, elke mijlpaal die je behaalt, en nu dan de promotie, als het nieuwe normaal aan te voelen. Gek genoeg is het gevoel van prestatie dat je verwachtte eigenlijk veranderd in een emotionele baseline, en veel van mijn doelen zijn verschoven of bleken achteraf gezien niet belangrijk. Niet om dit unieke moment naar beneden te halen, maar achteraf gezien ervaar ik mijn volledige opleiding, tot de onderzoeker die ik nu ben, als een van de vele traptreden van het leven in plaats van het ultieme doel dat in de voorgaande jaren in mijn hoofd werd geschetst. Ik ben vooral dankbaar voor de belangrijke lessen en ervaringen die ik heb kunnen opdoen, de kennis die ik in mijn vakgebied heb kunnen absorberen (deels ook door mijn obsessie met dit onderwerp) en de vele collega's met wie ik te maken heb gehad en die invloed hebben gehad op mijn ontwikkeling.

Allereerst een dankwoord aan mijn familie voor de ondersteuning en het vangnet dat jullie mij boden. Jullie weten wie ik bedoel, dus ik bespaar jullie hier van cringy opmerkingen.

Ik noem hieronder in chronologische volgorde de extra speciale mensen.

Als eerste bedank ik Prasanna Gururajan en Victor Schuurmans, die mij in het afstudeerjaar van mijn bachelor enorm veel ruimte, optimisme en kansen boden voor mijn ontwikkeling. Dit was eigenlijk het jaar waarin ik de machine learning en reinforcement learning velden indook, en de expertise die ik toen tijdens mijn stage heb opgedaan, heb ik tot op de dag van vandaag profijt.

Een enorm dankjewel aan Aske Plaat, die nu in mijn PhD-commissie zit, maar mij ook tijdens de master begeleidde. Eigenlijk heb ik (natuurlijk ook door mijn eigen inzet) de PhD-positie grotendeels aan hem te danken, omdat hij en Thomas Moerland mij aan mijn nieuwe supervisor hadden verkocht. Dus ook dankjewel aan Thomas Moerland voor de succesvolle begeleiding van mijn masterscriptie in de gekke coronatijd.

Dan mijn promotoren, Mathijs en Matthijs: bedankt voor jullie geduld, de ruimte die jullie mij hebben gegeven en dat jullie mijn ontwikkeling vooropstelden in dit lange PhD-traject. Op sommige momenten was ik koppig, op andere momenten slordig, maar ik hoop dat we het er nu allemaal over eens zijn dat het op een goede wijze is afgerond.

Dankjewel aan Willi Gottstein, die mij in het begin vanuit DSM-Firmenich begeleidde, maar later voor zijn droombaan vertrok. Volgens mij waren wij het methodologisch vaak niet eens over mijn onderzoek, maar ik denk dat we beiden wat hebben geleerd, hoogstwaarschijnlijk iets verschillends. Een speciaal dankjewel aan Stijn Bierman, die zonder context als vervanger van Willi vanuit DSM-Firmenich in mijn begeleiding stapte.

Je stond open voor waar ik mee bezig was, stelde mijn ontwikkeling en PhD voorop, en je bent toegankelijk. Dus enorm bedankt dat je deze rol hebt willen oppakken.

Dan nog een dankjewel aan de commissie voor hun tijd bij het beoordelen van mijn proefschrift en het bijwonen van de verdediging.

Aan mijn collega's: dankjewel aan de SDM-groep en de Algorithmics-groep, jullie maakten de omgeving die ik de afgelopen vier jaar mijn kantoor heb kunnen noemen. Dankjewel aan de collega's van de AI4b.io-samenwerking, wat ook nuttig was. Dankjewel aan Daniël Vos: je hebt droge humor en bent een betrokken persoon, en ik weet nog steeds niet zeker of je af en toe met je dubbelganger wisselde. Dankjewel aan Moritz Zanger voor je enorme toewijding om de reading group levend te houden. Dankjewel aan Oussama Azizi, die vet grappige memes stuurt, we hebben elkaars Insta. Dankjewel aan de whizzkid Pascal, die altijd slimme Bayesian opmerkingen klaar had staan.

Een speciaal dankjewel aan Jinke He: zonder jou had ik geen papers en waarschijnlijk geen proefschrift. Een speciaal dankjewel aan Yaniv Oren: jouw zelfovertuiging straalt af op anderen, dus ik denk dat ik nu ook in mijzelf geloof. Jinke & Yaniv, dankje voor jullie optimisme, enthousiasme en betrokkenheid wat ik nodig had in de lastigere fases van het onderzoek.

Ik ben het Echo-gebouw van de TU Delft niet dankbaar, vanwege de vele en continue technische kwalen, het omgekeerde ontwerp voor het gebruik als kantoor, en het collectieve 'analysis-paralysis' van de beheerders om ook maar iets op te lossen. Als genoeg managers een rapport schrijven is uiteindelijk niemand verantwoordelijk. Maar hé, het staat wel leuk op de voorpagina van de krant.

Joery A. de Vries
Alphen aan den Rijn, augustus 2026

Curriculum Vitæ

Joery Ariën de Vries
linkedin.com/in/joery-de-vries/

29-08-1997	Born in Woerden, the Netherlands.

Education

2015–2019	BSc. Bioinformatics Leiden University of Applied Sciences
2019–2021	MSc. Computer Science: Data Science Leiden University
2021–2026	PhD. Computer Science Delft University of Technology

Work Experience

2018–2019	Data Analyst & Research Intern Janssen Pharmaceutical Companies of Johnson & Johnson
2026–	AI Researcher Trent AI Limited

List of Publications

5. de Vries, J. A., He, J., Oren, Y., van der Vaart, P. R., de Weerdt, M. M., & Spaan, M. T. J. (2026). VariBASed: Variational Bayes-Adaptive Sequential Monte-Carlo Planning for Deep Reinforcement Learning. arXiv preprint arXiv:2602.18857

4. Oren, Y., de Vries, J. A., van der Vaart, P. R., Spaan, M. T. J., & Böhmer, W. (2025). Parallelizing Tree Search with Twice Sequential Monte Carlo. arXiv preprint arXiv:2511.14220

3. de Vries, J. A., He, J., Oren, Y., & Spaan, M. T. J. (2025). Trust-Region Twisted Policy Improvement. *Forty-second International Conference on Machine Learning*

2. de Vries, J. A., He, J., de Weerdt, M., & Spaan, M. T. J. (2025). Bayesian Meta-Reinforcement Learning with Laplace Variational Recurrent Networks. *Reinforcement Learning Conference*

1. He, J., Moerland, T. M., de Vries, J. A., & Oliehoek, F. A. (2024). What model does MuZero learn? In U. Endriss, F. S. Melo, K. Bach, A. J. B. Diz, J. M. Alonso-Moral, S. Barro, & F. Heintz (Eds.), *ECAI 2024 - 27th European Conference on Artificial Intelligence,* (pp. 1599–1606, Vol. 392). IOS Press. https://doi.org/10.3233/FAIA240666

Contributed to

1. van Leeuwen, S. A. J. (2023). *Language Assistance in Reinforcement Learning in Dynamic Environments* [Master's thesis, Delft University of Technology]. TU Delft Repository

List of Repositories

1. https://github.com/joeryjoery/jit_env

2. https://github.com/joeryjoery/trtpi

3. https://github.com/joeryjoery/lvrnn

See also the associated electronic collection of Software at:
https://doi.org/10.4121/cb719326-75c8-411f-a26d-2cdb6faceef7