University of Liège
School of Engineering
Montefiore Institute

Reinforcement Learning in
Partially Observable Markov Decision Processes

Learning to Remember the Past by Learning to Predict the Future

Gaspard Lambrechts

Advisors
Prof. Damien Ernst
Prof. Guillaume Drion

Jury

Prof. Pierre Geurts, Université de Liège, President.
Prof. Damien Ernst, Université de Liège, Advisor.
Prof. Guillaume Drion, Université de Liège, Advisor.
Prof. Aditya Mahajan, McGill University.
Prof. Vincent François-Lavet, Vrije Universiteit Amsterdam.
Prof. Matthijs Spaan, Technische Universiteit Delft.
Prof. Gilles Louppe, Université de Liège.
Dr. Raphaël Fonteneau, Université de Liège.

Summary

Intelligence is usually understood as the ability to make decisions, based on perception, in order to achieve objectives. In other words, intelligence is about perceiving and abstracting past information about the world for then acting on its future execution. This thesis focuses on reinforcement learning in partially observable Markov decision processes for learning intelligent behaviors through interaction. In particular, this manuscript explores and emphasizes the interplay between perception, representations, memory, predictions and decisions. After introducing the theoretical foundations, the core contributions of the thesis are presented across three thematic parts.

The first part, "Learning and Remembering," investigates how learning intelligent behaviors improves memory and vice versa. To begin with, it studies how learning to act optimally results in representations of the perception history that encode the posterior distribution over the states, known as the belief. Next, it studies how long-term memory improves the ability to learn intelligent behaviors, by designing an initialization procedure for recurrent neural networks that endows them with long-term memorization abilities.

The second part, "Leveraging Additional Information," explores how additional information about the world can be used to learn intelligent behaviors faster than when learning from perception only. It starts by empirically showing that world models predicting this additional information provide better history representations and faster learning. Then, it provides a theoretical justification for the improved convergence speed of a particular algorithm that leverages this information, namely the asymmetric actor-critic algorithm.

The third part, "Entangling Predictions and Decisions," proposes several architectural innovations for obtaining world models that efficiently generate trajectories. First, it develops new sequence models that parallelize autoregressive generation, while being implicitly recurrent to allow resuming generation. Afterwards, it elaborates on their use as new world models that are able to generate trajectories in parallel through specific latent policies.

Finally, this thesis concludes by summarizing how learning adequate representations of the perception history is paramount to learning to make decisions under partial observability. In the perspective of developing general intelligence, this thesis also motivates the shift from specialized abstractions to generalizable abstractions extending across diverse environments.

Contributions

This thesis is organized around the following papers.

- Part I. Learning and Remembering.
  - Chapter 3. Learning for Remembering.
    Gaspard Lambrechts, Adrien Bolland, and Damien Ernst. Recurrent Networks, Hidden States and Beliefs in Partially Observable Environments. *Transactions on Machine Learning Research*, 2022.
  - Chapter 4. Remembering for Learning.
    Gaspard Lambrechts, Florent De Geeter, Nicolas Vecoven, Damien Ernst, and Guillaume Drion. Warming Up Recurrent Neural Networks to Maximise Reachable Multistability Greatly Improves Learning. *Neural Networks*, 2023.
- Part II. Leveraging Additional Information.
  - Chapter 5. Remembering by Predicting Additional Information.
    Gaspard Lambrechts, Adrien Bolland, and Damien Ernst. Informed POMDP: Leveraging Additional Information in Model-Based RL. *Reinforcement Learning Journal*, 2024a.
  - Chapter 6. Learning Faster with Additional Information.
    Gaspard Lambrechts, Damien Ernst, and Aditya Mahajan. A Theoretical Justification for Asymmetric Actor-Critic Algorithms. *International Conference on Machine Learning*, 2025.
- Part III. Entangling Predictions and Decisions.
  - Chapter 7. Rolling the Dice First.
    Gaspard Lambrechts, Yann Claes, Pierre Geurts, and Damien Ernst. Parallelizing Autoregressive Generation with Variational State Space Models. *ICML Workshop on the Next Generation of Sequence Modeling Architectures*, 2024b.

These additional papers are not discussed in this thesis.

- Adrien Bolland, Gaspard Lambrechts, and Damien Ernst. Behind the Myth of Exploration in Policy Gradients. *arXiv:2402.00162*, 2024a.
- Arthur Louette, Gaspard Lambrechts, Damien Ernst, Eric Pirard, and Godefroid Dislaire. Reinforcement Learning to Improve Delta Robot Throws for Sorting Scrap Metal. *arXiv:2406.13453*, 2024.
- Adrien Bolland, Gaspard Lambrechts, and Damien Ernst. Off-Policy Maximum Entropy RL with Future State and Action Visitation Measures. *arXiv:2412.06655*, 2024b.

Acknowledgments

First, I would like to thank my advisor, Damien Ernst, for his supervision, but also for encouraging independence, for the opportunities he provided, and for teaching me his rigorous approach to writing. I also thank Guillaume Drion for the numerous interesting discussions that we had. Finally, I am grateful to Aditya Mahajan for welcoming me to McGill University in Montréal, and for all the things he taught me. Obviously, I thank the jury for their careful reading of the thesis and for their interesting feedbacks. I also gratefully acknowledge the financial support of the FNRS for my FRIA grant.

Next, I would like to thank my most faithful coauteur, Adrien Bolland, who has contributed significantly to this thesis, from whom I learned a lot, and with whom I had a lot of interesting research discussions.

I am grateful to Pierre Geurts, Louis Wehenkel and Gilles Louppe for the fascinating courses they teach at the University of Liège. Along with Raphaël Fonteneau and Vân Anh Huynh-Thu, I appreciate their approachability and the enjoyable discussions we had during lunch breaks.

I would also like to thank all my other coauthors: Florent De Geeter, Nicolas Vecoven, Arthur Louette and Yann Claes.

In particular, I would like to thank Yann Claes for sharing my office during these four years, and for the great atmosphere in the legendary I.106 office of the Montefiore Institute. I would also like to thank my colleagues Pascal Leroy and Matthias Pirlet from the office across the hall, for all the coffee breaks and the good vibes. This journey would have been very different without them.

I would like to thank my old friend Victor Dachet for his good mood and presence throughout my thesis and beyond, Louis Dupont for the moments we spent together playing board games or bouldering, and my friend David Martin for all the nice discussions we had about research and other topics.

I also want to thank the other colleagues present at our breaks, in particular Renaud Vandeghen and Adrien De Voeght. I would like to extend my thanks to my former colleagues Arnaud Delaunoy, Antoine Dubois and Antoine Wehenkel, who also contributed to the great times at the office. Finally, I thank all the reinforcement learners for the nice research discussions, which includes Maurizio Vassallo, Samy Mokeddem, Laurie Boveroux and Daniel Ebi.

I would then like to thank all the other people from Montefiore who makes it so special to work in this great atmosphere, especially the researchers that we

usually see at the aquarium, and all the people from the smart grid lab.

For all the great times we spent together over the last few years, I would like to thank my friends Marie Crutzen and Anouck Petit, but also my old friends Clara Mativa, Clara Troquet and Sarah Salpetier.

I also want to thank my old flatmates Boris Martin and Bastien Ewbank, for the nice evenings we shared for a couple of years.

I also thank the technical staff of Montefiore, in particular Denis Bourguignon, Éric Vangenechten, Sophie Cimino, Patricia Potier and Khadija Nid-Lhint.

Finally, I would like to thank my family for their support and for all the good things that they inspire me. I would especially like to thank my parents, sisters, and grandparents for all the love they have given me, for having always pushed me to cultivate my curiosity, and for always being interested.

And most of all, I thank Elif Sakalihasan for her love and support, day in, day out. While I was "learning to remember the past by learning to predict the future," she took care of reminding me to also enjoy the present moment. I am thankful to her for accompanying me and for her constant attention throughout this thesis, which I dedicate to her.

Contents

List of Theorems

List of Algorithms

List of Tables

List of Figures

Introduction

Chapter 1

A Matter of Perception

Intelligence is usually understood as the ability, and in particular the computational ability, to make decisions in order to achieve objectives [McCarthy, 1998]. This definition probably overlooks, but certainly encompasses, the operation of processing the available information for then forming a decision. Indeed, decision making does not emerge spontaneously but is rather initiated by perception, which serves as the stimulus and basis upon which the decision is constructed. From the processed perception, the decision may then be derived through either a learned computation or a planning procedure, that both seek to optimize the outcomes for the objective at hand. We find useful to establish a permeable distinction between the process of abstracting perception into a representation of that information, and the computation of a decision based on this representation. This separation sheds light on very different aspects of intelligent decision making: those concerned with the adequate processing and representation of past information, and those concerned with the adequate computation of decisions for achieving desired outcomes. In short, we frame intelligence as the ability of perceiving and abstracting past information about the world for then acting on its future execution, in the perspective of achieving an objective.

This thesis is interested in learning such intelligent behaviors, mapping past and present perception of the world to immediate and future actions. To that hand, we rely on reinforcement learning (RL) in partially observable Markov decision processes (POMDP) for learning these behaviors through interaction. First, let us put aside the learning process, and let us focus on the solution of the problem of optimally controlling a POMDP. This formalization of decision processes assumes an underlying Markovian state that is unobservable, but from which we get partial observations. In other words, it assumes that the perception of the world is partial, and that we are not provided with its full state. A consequence of this partial perception is the need of remembering some information from past observations, to infer relevant information about the state for making an optimal decision. It contrasts with the usual but more restrictive notion of fully observable Markov decision process (MDP) that assumes the state to be fully observable. Both models assume a discrete decision scheme, where actions are successively taken, based on the history of observations and past actions, to influence future states time step after time step. These formalizations also assume

a reward process such that a scalar reward is provided for each action taken in the decision process. This relies on the reward hypothesis, which states that any desired behavior can be formulated as a reward maximization problem [Sutton and Barto, 1998]. Given a POMDP, the decision making problem becomes that of selecting actions to maximize rewards, based on the history of observations and past actions. In the next two paragraphs, we review lessons from optimal control theory, which inform us about the optimal solution to this problem, but also about the RL methods to be applied, which are developed further below.

As far as the abstraction of perception is concerned, the objective is to process the stream of observations to extract relevant information for acting at all future time steps, and to discard the rest. In other words, the crucial aspect of this perception process is to memorize relevant information for acting now, but also for processing future observations and acting optimally in the future. The optimal control theory clearly established the sufficiency of the posterior state estimation, known as the belief, for optimally controlling a POMDP [Åström, 1965]. The belief is defined as the posterior distribution over the states given the history of observations and past actions. There thus exists a statistic of the history, called the belief, that is sufficient to be considered for acting optimally now, and also sufficient to be memorized for processing future observations and acting optimally at all later time steps. As a corollary, the theory states the existence of sufficient statistics for optimal control, which may not necessarily be the belief [Striebel, 1965]. In particular, when relaxing the POMDP model by not assuming an underlying state, their may still exist sufficient statistics for optimal control, which are sometimes called information states in that more general setting [Bertsekas, 2012]. In the case of MDPs, the state is known to be a sufficient statistic, which intuitively follows from its Markovian property, stating that it perfectly summarizes the past to predict the future.

As far as the selection of an optimal action for the future is concerned, optimal control theory studied the problem using dynamic programming [Bellman, 1952] in the case of MDP [Bellman, 1957]. Dynamic programming relies on the principle of optimality, which applies to any MDP in the sense that the solution has an optimal substructure. In other words, the optimal sequence of actions can be decomposed into the problem of selecting an optimal action now, and the problem of selecting the optimal sequence of future actions at the next time step. Through this recursive structure, the general problem embeds the smaller problem of optimally controlling the MDP for the remaining time. It enables the application of dynamic programming for solving the optimal control problem of an MDP. Roughly speaking, this algorithm works backward and computes an optimal action for maximizing the next reward, from which it computes an optimal action for maximizing the next two rewards, and so on. Thanks to the previously mentioned findings that established the sufficiency of the belief for optimally controlling a POMDP, dynamic programming extends to POMDPs. Indeed, because the belief summarizes all information from the history for predicting the immediate reward, as well as for predicting future beliefs given future actions, it is the Markovian state of an equivalent MDP that we call the belief MDP. It is however worth noting that the belief MDP is a particular case of MDP, in the sense that its optimal solution has a particular form, which is piecewise linear and convex in its state, the belief [Smallwood and Sondik, 1973]. The transformation of a POMDP into an MDP through the

belief filter once again highlights the distinction that exists between the problem of processing the past on the one hand, and the problem of planning for the future on the other hand.

As will be explicitly developed in the next part of this thesis that covers the mathematical background, these results from optimal control theory are insightful about the structure of the solution for optimally controlling a POMDP. However, the computation of these solutions is usually computationally intractable. First, the belief computation requires to integrate over the state space at every time step, and to store the distribution over the state space, both of which can become difficult when the state space is large, or impossible when it is continuous. Similarly, each step of the dynamic programming computation requires to integrate over the state space, and to store the solution for every state, which poses the same problems with large state spaces. In particular, for a POMDP with finite state space, the state space of the equivalent belief MDP is infinite. Fortunately, the particular structure of the solution of the belief MDP, which is piecewise linear and convex, still allows the execution of the dynamic programming algorithm. However, other challenges arise from the representation and evaluation of the solution, which grows exponentially with the number of steps of the dynamic programming algorithm. All these considerations make the problem of optimally controlling a POMDP challenging at best, impossible at worse. Last, but not least, the applicability of the dynamic programming algorithm relies on the assumption that the POMDP model is known, that is the state space, action space, observation space, along with the exact transition distribution, observation distribution and reward distribution. While some problems may satisfy this hypothesis, we relax this assumption and focus on learning intelligent behaviors in any POMDP, from interaction only.

For this purpose, this thesis focuses on RL in POMDP [Spaan, 2012]. The RL approach is appealing for solving, or approximately solving, decision making problems, notably because it makes very few assumptions about the problem at hand. In its purest form, the promise of an RL algorithm is to learn an optimal behavior from interaction with an environment whose dynamics are unknown. More formally, an RL algorithm aims at learning a policy, which is defined as a mapping from states or histories to actions, in order to maximize the reward signal, and that from samples obtained by interacting with an environment. The maximization of the reward signal is usually implemented by one of the following criteria: the maximization of the expected sum of rewards over a finite horizon, the maximization of the expected average reward over the infinite horizon, or the maximization of the expected sum of discounted rewards over the infinite horizon. In the following, we refer to any of these objectives as the return. Let us review the most popular and effective RL approaches, which were initially developed in the fully observable setting. First, policy-gradient methods primarily work with a parametrized model of the policy, for which an unbiased estimate of the gradient of the return can be computed from samples of interaction [Williams, 2004]. As a results, these methods directly optimize the return of the policy using stochastic gradient ascent on their parameters. It is also worth mentioning the so-called actor-critic approaches, which are policy-gradient methods that reduce the variance of the gradient estimate by jointly learning a value function. This value function estimates the return of the policy being optimized, which is substituted to the empirical return in the computa-

tion of the policy gradient estimate [Sutton et al., 1999]. Second, value-based methods primarily work with a model of the value function, which directly estimates the optimal return that can be obtained from a state, from which a near-optimal policy can be derived. This is achieved through a learning process akin to stochastic incremental dynamic programming [Watkins, 1989], using samples obtained from interaction. Third, model-based methods primarily work with a model of the environment that is learned in an unsupervised learning fashion for then deriving an optimal policy. This can be achieved in two main ways: through the application of any of the two previously mentioned RL approaches, or through the application of optimal control to this learned model. The latter approach is similar the traditional pipeline of system identification and optimal control [Åström and Eykhoff, 1971].

Despite these RL methods being initially designed for learning to optimally control a MDP from interaction, they were straightforwardly adapted to the optimal control of a POMDP. Without any model of the decision process based on which to compute the belief, the delineation between the problem of abstracting past perception and the problem of computing the optimal action is blurred. This characterizes the coupled burden of RL in POMDP, which is to jointly learn to process the history of observations and past actions, and to produce optimal actions for the future execution of the decision process. In practice, the RL field has historically mostly ignored this distinction and has completely coupled these aspects, by considering the complete history as the Markovian state of an equivalent MDP that we call the history MDP. This rather crude approach, which completely ignores the structure of the solution and the unbounded growth of the history, has proven quite effective in practice. We identify the following seminal works in history-dependent RL, for each of the three aforementioned approaches. In policy-gradient approaches, history-dependent policies were proposed, using recurrent neural network to process these variable-length histories [Wierstra et al., 2007]. In value-based approaches, history-dependent value functions were proposed using recurrent neural networks as well [Bakker, 2001]. In model-based approaches, history-dependent models of the environments were proposed, once again using recurrent neural networks [Lin and Mitchell, 1992]. These early works had paved the way for adapting to the partially observable setting the numerous RL breakthroughs enabled by the deep learning revolution, in policy-gradient approaches [Heess et al., 2015], value-based approaches [Hausknecht and Stone, 2015], and model-based approaches [Ha and Schmidhuber, 2018]. While these methods starts from the history MDP, by noticing that the history can be simply substituted to the state in traditional RL methods, it is worth noting that they process the history with a recurrent neural network, which compresses the history into a statistic. As a results, it becomes a requirement for this statistic of the history to be sufficient for optimal control.

Although these methods are theoretically able to learn sufficient statistics of the history for the optimal control of any POMDP, sufficient statistics of the history can also be learned explicitly. Early approaches proposed to learn statistics of the history that are predictive of future observations, which was called predictive state representations [Littman and Sutton, 2001]. The idea of learning a recurrently updatable statistic that is predictive of the observations of the decision process roughly amounts to assuming and learning a generative model for the decision process. This illustrates the strong ties that exist between the

problem of learning to abstract past perception into representations, and the problem of predicting future execution. In deep RL, an important line of work has considered auxiliary representation learning objectives for the hidden state of recurrent policies or recurrent value functions, in order to ensure the learning of a sufficient statistic [Igl et al., 2018, Guo et al., 2018, Gregor et al., 2019]. Moreover, because these representation objectives offer a predictive model of the environment, history-dependent model-based methods were simply developed as an effective approach for RL in POMDP, offering these representation objectives for free [Hafner et al., 2019, Samsami et al., 2024]. In other words, model-based RL completely encompasses usual representation learning objectives for the abstraction of perception in partially observable environments.

To close this overview of RL in POMDP, we want to take a step back and reflect on the constraints we have imposed so far. While it is realistic to assume a partial perception of the state of the decision process, it may be too pessimistic to assume the same partial perception of the state while learning. It is indeed strictly more general to assume that additional information about the state may be available while learning, and it is certainly interesting to develop methods that leverage this eventual additional information. This observation has resulted in a recent line of work called asymmetric learning, or asymmetric RL, whose name refers to the asymmetry of observability between learning and execution. Early approaches notably proposed to imitate a privileged policy conditioned on the state [Choudhury et al., 2018], or to use an asymmetric critic conditioned on the state [Pinto et al., 2018]. These heuristic methods initially lacked a theoretical framework, and a recent line of work has focused on proposing theoretically grounded asymmetric learning objectives. First, imitation learning of a privileged policy was known to be suboptimal, and it was addressed by constraining the privileged policy so that its imitation results in an optimal policy for the partially observable environment [Warrington et al., 2021]. Similarly, asymmetric actor-critic approaches were proven to provide biased gradients, and an unbiased actor-critic approach was proposed by introducing the history-state value function [Baisero and Amato, 2022]. In model-based RL, several works proposed world model objectives that are proved to provide sufficient statistics of the history, by leveraging the state [Avalos et al., 2024] or arbitrary state information [Lambrechts et al., 2024a]. Finally, asymmetric representation learning approaches were proposed to learn sufficient statistics using state samples [Wang et al., 2023, Sinha and Mahajan, 2023].

This thesis humbly contributes to this literature by analyzing and easing the burden of RL in POMDP, which is to jointly learn to perceive, represent, memorize, predict and decide. In Part I, we seek to understand the role of memory when learning to act in a decision process, which we articulate around two scientific papers. The first paper, "*Recurrent Network, Hidden States and Beliefs in Partially Observable Environments*," investigates the link between the memory resulting from the process of learning to act optimally, and the optimal statistic of the history that the optimal control theory prescribes, which is the belief. The second paper, "*Warming Up Recurrent Neural Networks to Maximize Reachable Multistability Greatly Improves Learning*," studies the importance of a good memorization ability as a prerequisite for learning to act optimally. In Part II, motivated by the findings of the previous papers, we seek to improve the learning of optimal behaviors by fostering a memory which encodes a suf-

ficient statistic, and this by leveraging eventual additional information during learning. The first paper, "*Informed POMDP: Leveraging Additional Information in Model-Based RL*," formally relaxes the POMDP assumptions to account for eventual additional information available during learning, and proposes a method that leverages it to learn a sufficient statistic of the history for optimal control. The second paper, "*A Theoretical Justification for Asymmetric Actor-Critic Algorithms*," aims at establishing a theoretical justification for a class of algorithms that leverage such additional information during learning, to better estimate the long term outcomes of the policy. In Part III, we explore topics where the entanglement of memory, predictions and decisions might be mutually beneficial. The short paper, "*Parallelizing Autoregressive Generation with Variational State Space Models*," proposes a new sequence modeling architecture that combines the desirable properties of many previously proposed architectures, which are the parallelizability of history processing, the parallelizability of history generation, but also the implicit recurrence of these processes, so as to be able to resume generation without reprocessing all past history. In a last chapter, we elaborate on a possible usage of such sequence models in the context of model-based RL, where specific latent policies could unlock parallel generation of trajectories for learning efficiently.

The conclusions offered by this research project are plural. Most importantly, we think that this thesis establishes a clear motivation for the particular attention that should be given to the problem of learning to abstract perception into useful representations. It notably highlights the limits of disregarding the structure of the solution with the history MDP, and motivates the consideration of that structure with representation learning. More precisely, this thesis motivates to learn abstractions of the world that are predictive of its future execution, as a representation learning method that offers good representations for decision making. We think that such approaches correctly considers the distinction between the two different aspects of decision making, processing the past and planning for the future, while not separating them. Indeed, history-dependent RL with representation learning makes a permeable distinction that allows these tasks to inform each other, and interact in a mutually beneficial way. Finally, we also want to reflect on the fact that in this introduction we have presented RL as a method for solving decision making in POMDP, or more precisely, in any POMDP taken separately. Many researchers like to see RL as a problem rather than a method, the problem of learning to make decisions, generally speaking. In other words, the RL problem is to learn to act optimally and generalize over a distribution of POMDPs. Magnificently, the problem of learning to generalize over a distribution of POMDPs can be framed as a POMDP. As a future work, this thesis thus also motivates the consideration of RL in POMDP to solve the RL problem, which is to develop generalizable intelligence.

Background

Chapter 2

Reinforcement Learning under Partial Observability

2.1 Notations and Conventions

This manuscript makes use of the notations detailed in Table 2.1. We notably denote random variables with uppercase letters and their realizations with lowercase letters. We also index these random variables and their realizations with time to denote random processes and their realizations. For brevity, we sometimes denote a random variable from a random process at an arbitrary time by omitting its time index, and we denote a random variable from the same random process at the next time step with prime. We denote the set of real numbers, integer numbers, and natural numbers with their usual symbols. Finally, given a discrete space $\mathcal{X}$, we denote by $2^{\mathcal{X}}$ the discrete σ-algebra, or power set, and we use $\Delta(\mathcal{X})$ to denote the set of probability measures over the measurable space $(\mathcal{X}, 2^{\mathcal{X}})$. Given a continuous space $\mathcal{X}$, we denote with $\mathcal{B}_{\mathcal{X}}$ the Borel σ-algebra, and we use $\Delta(\mathcal{X})$ to denote the set of probability measures over the measurable space $(\mathcal{X}, \mathcal{B}_{\mathcal{X}})$. Finally, we denote by δ_c the Dirac measure centered at c.

Depending on the context, we may denote the expectation of a random variable X with sample space $\mathcal{X}$ and probability measure P of density p, with any the following notations,

$$\mathbb{E}[X] = \mathbb{E}_P[X] = \mathop{\mathbb{E}}_{p(x)}[x] = \mathop{\mathbb{E}}_{x \sim p(\cdot)}[x] = \mathop{\mathbb{E}}_{x \sim P}[x] = \int_{\mathcal{X}} x p(x) \, \mathrm{d}x = \int_{\mathcal{X}} x \, \mathrm{d}P. \quad (2.1)$$

Following the reinforcement learning literature, we denote the expectation of a random variable at a time step t under the distribution induced by a policy π with $\mathbb{E}^{\pi}[X_t]$. Similarly, we denote the expectation of a random variable under the discounted visitation measure $d^{\pi,\gamma}$ induced by a policy π with $\mathbb{E}^{d^{\pi,\gamma}}[X]$. These mathematical objects are clearly defined in the next sections.

Notation	Meaning
$\mathcal{X}$	Set.
X	Random variable.
x	Realization of a random variable.
X_t	Random variable from a given random process at time step t.
x_t	Realization of a random variable from a given random process at time step t.
$X_{k:t}$	Sequence of random variables from a given random process from time step k to time step t, both inclusive.
$x_{k:t}$	Sequence of realizations of random variables from a given random process from time step k to time step t, both inclusive.
X, X'	Random variables from a given random process at arbitrary time steps t and $t+1$.
x, x'	Realizations of random variables from a given random process at arbitrary time steps t and $t+1$.
$\mathbb{R}$	Set of real numbers.
$\mathbb{Z}$	Set of integer numbers.
$\mathbb{N}$	Set of natural numbers.
$\mathbb{N}_0$	Set of natural numbers and zero.
$2^{\mathcal{X}}$	Power set over the discrete space $\mathcal{X}$.
$\mathcal{B}_{\mathcal{X}}$	Borel set over the continuous space $\mathcal{X}$.
$\Delta(\mathcal{X})$	Set of probability measures over $(\mathcal{X}, 2^{\mathcal{X}})$ or $(\mathcal{X}, \mathcal{B}_{\mathcal{X}})$.
$\delta_c(x)$	Dirac distribution centered at c.

Table 2.1: Notations and conventions in this manuscript.

2.2 Markov Decision Processes

Sequential decision making under full observability can be modeled as a Markov decision process (MDP). Formally, an MDP is defined as a tuple $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, R, P, \gamma)$ where,

- $\mathcal{S}$ is the state space,
- $\mathcal{A}$ is the action space,
- $T\colon \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ is the transition distribution,
- $R\colon \mathcal{S} \times \mathcal{A} \to \Delta(\mathbb{R})$ is the reward distribution,
- $P \in \Delta(\mathcal{S})$ is the initial distribution,
- $\gamma \in [0, 1)$ is the discount factor.

The initial state distribution P gives the probability $P(s_0)$ of $s_0 \in \mathcal{S}$ being the initial state of the decision process. The dynamics are described by the transition distribution T that gives the probability $T(s_{t+1}|s_t, a_t)$ of $s_{t+1} \in \mathcal{S}$ being the state resulting from taking action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$. The reward distribution R gives the probability density $R(r_t|s_t, a_t)$ of the immediate reward $r_t \in \mathbb{R}$ after taking action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$.[1] We also define the expected

[1] Some of the papers presented in this thesis will instead assume a deterministic reward function of the form $r_t = R(s_t, a_t, s_{t+1})$. These formalizations can be shown to be equivalent.

immediate reward as $\bar{r}_t = \overline{R}(s_t, a_t) = \mathbb{E}[R_t | S_t = s_t, A_t = a_t]$. Finally, the discount factor $\gamma \in [0, 1)$ weighs the relative importance of future rewards. The key assumption of an MDP is that states satisfy the Markov property,

$$\Pr(s_{t+1}|s_0, a_0, \ldots, s_t, a_t) = \Pr(s_{t+1}|h_t, a_t) = \Pr(s_{t+1}|s_t, a_t) = T(s_{t+1}|s_t, a_t). \tag{2.2}$$

Taking a sequence of t actions in an MDP conditions its execution and provides the observable history $h_t = (s_0, a_0, \ldots, s_t) \in \mathcal{H}_t \subset \mathcal{H}$, where $\mathcal{H}_t = (\mathcal{S} \times \mathcal{A})^t \times \mathcal{S}$ is the set of histories of size t, and $\mathcal{H} = \bigcup_{t=0}^{\infty} \mathcal{H}_t$ is the set of histories of arbitrary length. At any time $t \geq 0$, the current history h_t includes all information that is available to select action $a_t \in \mathcal{A}$. We define a history-dependent policy $\eta \in H$ as a mapping from histories to probability distributions over actions, where $H = \mathcal{H} \to \Delta(\mathcal{A})$ is the set of all history-dependent policies. We use $\eta(a|h)$ to denote the probability of selecting action $a \in \mathcal{A}$ in history $h \in \mathcal{H}$. Finally, we define the return $J(\eta)$ of a history-dependent policy $\eta \in H$ as the expected sum of discounted rewards,

$$J(\eta) = \mathbb{E}^{\eta}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right]. \tag{2.3}$$

A history-dependent policy $\eta^* \in H$ is said to be an optimal history-dependent policy when it maximizes the return,

$$\eta^* \in \arg\max_{\eta \in H} J(\eta). \tag{2.4}$$

Let us finally define a stationary Markov policy $\pi \in \Pi$ as a mapping from states to probability distributions over actions, with $\Pi = \mathcal{S} \to \Delta(\mathcal{A})$ the set of all stationary Markov policies. We also define the return $J(\pi)$ of a stationary Markov policy $\pi \in \Pi$ as the expected sum of discounted rewards,

$$J(\pi) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right]. \tag{2.5}$$

A stationary Markov policy $\pi^* \in \Pi$ is said to be an optimal stationary Markov policy when it maximizes the return,

$$\pi^* \in \arg\max_{\pi \in \Pi} J(\pi). \tag{2.6}$$

It was proven that stationary Markov policies are sufficient for optimally controlling an MDP [Puterman, 1994]. In other words, an optimal stationary Markov policy $\pi^* \in \Pi$ performs as well as an optimal history-dependent policy $\eta^* \in H$,

$$\max_{\pi \in \Pi} J(\pi) = \max_{\eta \in H} J(\eta). \tag{2.7}$$

This results stems from the Markov property of MDPs, which states that future states of the MDP are independent of past states given the current state.

Finally, we define the value function $V^\pi : \mathcal{S} \to \mathbb{R}$ of a policy $\pi \in \Pi$ as the return starting from a state $s \in \mathcal{S}$,

$$V^\pi(s) = \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t \middle| S_0 = s \right]. \tag{2.8}$$

We note that $J(\pi) = \mathbb{E}[V^\pi(S_0)]$. We also define the Q-function $Q^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ of a policy $\pi \in \Pi$ as the return starting from a state $s \in \mathcal{S}$ and an action $a \in \mathcal{A}$,

$$Q^\pi(s,a) = \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t R_t \middle| S_0 = s, A_0 = a \right]. \tag{2.9}$$

It is interesting to note that $V^\pi(s) = \mathbb{E}^\pi[Q^\pi(S,A)|S = s]$.

2.3 Optimal Control in Markov Decision Processes

Let us define the optimal Q-function as the Q-function of an optimal policy,

$$Q(s,a) = \max_{\pi \in \Pi} Q^\pi(s,a) = Q^{\pi^*}(s,a). \tag{2.10}$$

The optimal Q-function is the unique fixed point of the following Bellman operator [Bellman, 1957],

$$Q(s,a) = \mathbb{E}\left[R + \gamma \max_{a' \in \mathcal{A}} Q(S',a') \middle| S = s, A = a \right] \tag{2.11}$$

$$= \overline{R}(s,a) + \gamma \mathbb{E}\left[\max_{a' \in \mathcal{A}} Q(S',a') \middle| S = s, A = a \right]. \tag{2.12}$$

The solution of this nonlinear system of $|\mathcal{S}| \times |\mathcal{A}|$ equations can be approximated to an arbitrary precision using dynamic programming, assuming that we know the expected immediate reward $\overline{R}(s,a)$ for any state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$. The dynamic programming algorithm, known as Q-iteration in that case, iteratively computes Q-functions with a growing finite-time horizon. More precisely, the algorithm starts from the Q-function Q_0 with a horizon of zero, which is $Q_0(s,a) = 0$, $\forall s \in \mathcal{S}$, $\forall a \in \mathcal{A}$. Then, it successively computes for $k > 0$,

$$Q_k(s,a) = \overline{R}(s,a) + \gamma \mathbb{E}\left[\max_{a' \in \mathcal{A}} Q_{k-1}(S',a') \middle| S = s, A = a, \right]. \tag{2.13}$$

The Bellman operator being γ-contractive with the optimal Q-function as its unique fixed point, this sequence of Q-functions converges towards the optimal Q-function [Banach, 1922]. In order to store the solution for every state and action pair, and in order to compute the expectation, the dynamic programming algorithm requires discrete state and action spaces.

2.4 Partially Observable Markov Decision Processes

While assuming the existence of an underlying state for a decision process may seem reasonable, assuming that it will be fully observable is not. In the general

case, decision making problems only offer a partial observation of the underlying state of the decision process. Sequential decision making under partial observability can be modeled as a partially observable Markov decision process (POMDP). Formally, a POMDP is a tuple $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, P, \gamma)$ where,

- $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, R, P, \gamma)$ is an MDP whose states are not observable,
- $\mathcal{O}$ is the observation space,
- $O \colon \mathcal{S} \to \Delta(\mathcal{O})$ is the observation distribution.

The observation distribution O gives the probability $O(o_t|s_t)$ to get observation $o_t \in \mathcal{O}$ in state $s_t \in \mathcal{S}$. Interestingly, observations in a POMDP do not satisfy any Markov property in the general case,

$$\Pr(o_{t+1}|o_0, a_0, \ldots, o_t, a_t) = \Pr(o_{t+1}|h_t, a_t) \neq \Pr(o_{t+1}|o_t, a_t). \tag{2.14}$$

Taking a sequence of t actions in a POMDP conditions its execution and provides the observable history $h_t = (o_0, a_0, \ldots, o_t) \in \mathcal{H}_t \subset \mathcal{H}$, where $\mathcal{H}_t = (\mathcal{O} \times \mathcal{A})^t \times \mathcal{O}$ is the set of histories of size t, and $\mathcal{H} = \bigcup_{t=0}^{\infty} \mathcal{H}_t$ is the set of histories of arbitrary length. Note that for discrete state space $\mathcal{S}$, discrete action space $\mathcal{A}$ and discrete observation space $\mathcal{O}$, the history space $\mathcal{H}$ is countable. Similarly to MDPs, we define a history-dependent policy $\eta \in H$ as a mapping from histories to probability distributions over actions, where $H = \mathcal{H} \to \Delta(\mathcal{A})$ is the set of all history-dependent policies. We also define the return $J(\eta)$ of a history-dependent policy $\eta \in H$ as the expected sum of discounted rewards,

$$J(\eta) = \mathbb{E}^{\eta}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right]. \tag{2.15}$$

A history-dependent policy $\eta^* \in H$ is said to be an optimal history-dependent policy when it maximizes the return,

$$\eta^* \in \arg\max_{\eta \in H} J(\eta). \tag{2.16}$$

Because we do not have the Markov property in POMDPs, stationary Markov policies that would be conditioned on the current observation only are not sufficient for optimal control [Littman et al., 1995]. As a result, we must rely on history-dependent policies for optimally controlling POMDPs.

We now define the value function $V^{\eta} \colon \mathcal{H} \to \mathbb{R}$ of a history-dependent policy $\eta \in H$ as the return starting from a history h,

$$V^{\eta}(h) = \mathbb{E}^{\eta}\left[\sum_{t=0}^{\infty} \gamma^t R_t \middle| H_0 = h\right]. \tag{2.17}$$

We note that $J(\eta) = \mathbb{E}[V^{\eta}(H_0)]$. Let us also define the Q-function $Q^{\eta} \colon \mathcal{H} \times \mathcal{A} \to \mathbb{R}$ of a policy $\eta \in H$ as the return starting from a history h and action a,

$$Q^{\eta}(h, a) = \mathbb{E}^{\eta}\left[\sum_{t=0}^{\infty} \gamma^t R_t \middle| H_0 = h, A_0 = a\right]. \tag{2.18}$$

It is interesting to note that $V^{\eta}(h) = \mathbb{E}^{\eta}[Q^{\eta}(H, A)|H = h]$.

2.5 Belief Markov Decision Processes

Given a POMDP $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, P, \gamma)$, we define the belief $b \in \mathcal{B} \subsetneq \Delta(\mathcal{S})$ of a history $h \in \mathcal{H}$ as the posterior distribution over the states given the history, where $\mathcal{B}$ is the set of all attainable beliefs. Formally, the belief $b = f(h)$ is,

$$b(s) = \Pr(s|h), \tag{2.19}$$

where $f\colon \mathcal{H} \to \mathcal{B}$ is called the belief filter. Using Bayes' rule, the initial belief $b_0 = f(h_0)$ is given by,

$$b_0(s_0) = \Pr(s_0|h_0) = \Pr(s_0|o_0) \tag{2.20}$$

$$= \frac{P(s_0)O(o_0|s_0)}{\sum_{s_0 \in \mathcal{S}} P(s_0)O(o_0|s_0)}. \tag{2.21}$$

At all later time steps, the belief filter admits a recursive form $b_{t+1} = f(h_{t+1}) = u(b_t, a_t, o_{t+1}) = u(f(h_t), a_t, o_{t+1})$ where the update u is given by,

$$b_{t+1}(s_{t+1}) = \Pr(s_{t+1}|h_{t+1}) = \Pr(s_{t+1}|h_t, a_t, o_{t+1}) \tag{2.22}$$

$$= \frac{O(o_{t+1}|s_{t+1}) \sum_{s_t} T(s_{t+1}|s_t, a_t) b_t(s_t)}{\sum_{s_{t+1} \in \mathcal{S}} O(o_{t+1}|s_{t+1}) \sum_{s_t} T(s_{t+1}|s_t, a_t) b_t(s_t)}. \tag{2.23}$$

The proof, which we attribute to Ho and Lee [1964] and which is a consequence of Bayes' rule [Bayes, 1763], is given in Appendix A.

Note that for discrete state space $\mathcal{S}$, discrete action space $\mathcal{A}$ and discrete observation space $\mathcal{O}$, the history space $\mathcal{H}$ is countable and, as a consequence, the belief space $\mathcal{B}$ is also countable.

Given a POMDP $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, P, \gamma)$, it is possible to define an equivalent MDP, which is called the belief MDP. The belief MDP $\mathcal{M}'$ is given by $\mathcal{M}' = (\mathcal{S}', \mathcal{A}, T', R', P', \gamma)$ where,

- $\mathcal{S}' = \mathcal{B}$ is the belief space,
- $T'\colon \mathcal{B} \times \mathcal{A} \to \Delta(\mathcal{B})$ is the belief transition distribution,
- $R'\colon \mathcal{B} \times \mathcal{A} \to \Delta(\mathbb{R})$ is the belief reward distribution,
- $P' \in \Delta(\mathcal{B})$ is the initial belief distribution.

The initial belief distribution P' over the countable set of beliefs is given by,

$$P'(b_0) = \sum_{o_0 \in \mathcal{O}} \delta_{f(h_0)}(b_0) \sum_{s_0 \in \mathcal{S}} O(o_0|s_0) P(s_0). \tag{2.24}$$

The transition distribution T' is given by,

$$T'(b_{t+1}|b_t, a_t) = \sum_{o_{t+1} \in \mathcal{O}} \delta_{u(b_t, a_t, o_{t+1})}(b_{t+1}) \sum_{s_{t+1} \in \mathcal{S}} O(o_{t+1}|s_{t+1}) \sum_{s_t \in \mathcal{S}} T(s_{t+1}|s_t, a_t) b_t(s_t). \tag{2.25}$$

The reward distribution R' is defined as,

$$R'(r_t|b_t, a_t) = \sum_{s_t \in \mathcal{S}} R(r_t|s_t, a_t) b_t(s_t). \tag{2.26}$$

We also define the expected immediate reward in the belief MDP as $\bar{r}_t = \overline{R}'(b_t, a_t) = \mathbb{E}[R_t | B_t = b, A_t = a]$. The proof for the expression of the initial belief distribution, belief transition distribution and belief reward distribution of the belief MDP, which we attribute to Åström [1965], is given in Appendix B.

2.6 Optimal Control in Belief Markov Decision Processes

Given the existence of an equivalent belief MDP $\mathcal{M}'$ for any POMDP $\mathcal{P}$, we study the solution for this particular MDP. We know that a stationary Markov policy exists for this MDP. We thus define a belief policy $\pi' \in \Pi'$ as a mapping from belief to distribution over actions, where $\Pi' = \mathcal{B} \to \Delta(\mathcal{A})$ is the set of all belief policies. We define the optimal belief Q-function as,

$$Q'(b, a) = \max_{\pi' \in \Pi'} Q^{\pi'}(b, a), \tag{2.27}$$

where $Q^{\pi'}(b, a)$ is defined as the belief Q-function of a belief policy π',

$$Q^{\pi'}(b, a) = \mathbb{E}^{\pi'}\left[\sum_{t=0}^{\infty} \gamma^t R_t \middle| B_0 = b, A_0 = a\right]. \tag{2.28}$$

As for classical MDPs, it can be shown that the belief Q-function is the unique fixed point of the following Bellman operator [Sondik, 1978],

$$Q(b, a) = \mathbb{E}\left[R + \gamma \max_{a' \in \mathcal{A}} Q(B', a') \middle| B = b, A = a\right] \tag{2.29}$$

$$= \overline{R}'(b, a) + \gamma \mathbb{E}\left[\max_{a' \in \mathcal{A}} Q(B', a') \middle| B = b, A = a\right]. \tag{2.30}$$

Because the belief space is infinite, it seems a priori impossible to represent the Q-function or to compute the expectation over the belief space, which would render dynamic programming infeasible. However, the particular structure of the belief MDP makes dynamic programming applicable.

Let us first note that any piecewise linear and convex function can be represented by the epigraph of a finite number of linear functions, and that any linear function in $\mathbb{R}^d$ can be represented by a vector of size d. As a result, any piecewise linear and convex function f_{pwlc} can be represented with a finite set A of vectors, which we called α-vectors,

$$f_{\text{pwlc}}(x) = \max_{\alpha \in A} \sum_{i=1}^{d} \alpha_i x_i, \tag{2.31}$$

where $x \in \mathbb{R}^d$ and $\alpha \in \mathbb{R}^d$ are vectors of size d. It can be demonstrated by induction that the Q-function with a finite-time horizon is piecewise linear and convex in the belief vector. Indeed, the Q-function with a finite-time horizon of zero $Q_0(b, a) = 0,\ \forall b \in \mathcal{B},\ \forall a \in \mathcal{A}$ is trivially linear, which is piecewise linear and convex. Then, it can be shown that for any Q-function Q_N with finite-time

horizon N that is piecewise linear and convex in the belief vector, the Q-function with finite-time horizon $N+1$,

$$Q_{N+1}(b,a) = \overline{R}'(b,a) + \gamma \mathbb{E}\left[\max_{a' \in \mathcal{A}} Q_N(B', a') \middle| B = b, A = a\right]. \tag{2.32}$$

is also piecewise linear and convex in the belief vector. The proof, that we attribute to Smallwood and Sondik [1973] and that also provides the practical procedure for updating the set of α-vectors, is given in Appendix C. It is worth noting that this proof only proves the convexity of the optimal Q-function, to which the sequence of Q-function converges. The piecewise linearity of the optimal Q-function is nevertheless not guaranteed in the general case, since the number of α-vectors tends to infinity as the horizon N tends to infinity.

2.7 Reinforcement Learning under Partial Observability

As discussed in the introduction of this thesis, both the computation of the belief and the application of dynamic programming can become intractable for large state spaces, or even impossible for continuous state spaces. Moreover, this algorithm requires the model of the POMDP to be known, which can be unrealistic in many applications. As a result, we want to learn optimal policies from samples of interaction only, which is the focus of the field of reinforcement learning (RL). Since the belief is not computable without the model of the POMDP, we leave the belief MDP aside, and directly work with history-dependent policies and Q-functions. More precisely, the objective of RL in POMDP is to find a near-optimal policy $\eta \in \mathcal{H}$ from samples $\{h_k, a_k, r_k, o_{k+1}\}_{k=0}^{K-1}$ obtained by interacting with the POMDP.

Let us define the optimal Q-function, or simply Q-function, as the Q-function of an optimal policy,

$$Q(h,a) = \max_{\eta \in H} Q^{\eta}(h,a) = Q^{\eta^*}(h,a). \tag{2.33}$$

Given the existence of the equivalent belief MDP, the history Q-function can also be defined as the composition of the belief filter and the belief Q-function,

$$Q(h,a) = Q'(f(h),a), \ \forall h \in \mathcal{H}, \ \forall a \in \mathcal{A}, \tag{2.34}$$

As a consequence, the history Q-function is also the unique fixed point of a Bellman operator,

$$Q(h,a) = \mathbb{E}\left[R + \gamma \max_{a' \in \mathcal{A}} Q(H', a') \middle| H = h, A = a\right] \tag{2.35}$$

$$= \overline{R}(h,a) + \gamma \mathbb{E}\left[\max_{a' \in \mathcal{A}} Q(H', a') \middle| H = h, A = a\right], \tag{2.36}$$

where the expected immediate reward is defined as $\overline{R}(h,a) = \overline{R}'(f(h),a)$. This observation motivates the consideration of standard RL techniques for MDPs, by simply substituting the history to the state. Since the space of histories

is infinite, these method consider function approximators for processing these variable-length sequences, usually recurrent neural networks. In the following, we review the simplest adaptations of the main RL approaches to history-dependent models using differentiable parametric function approximators, also known as neural networks.

Given a POMDP $\mathcal{P}$ from which we can obtain samples, history-dependent Q-learning suggests to update a parametrized approximation $Q_\theta(h, a)$ of the optimal Q-function. More precisely, based on a transition (h_k, a_k, r_k, o_{k+1}) obtained by interacting with the environment, history-dependent Q-learning use the following parametric Q-learning update,

$$\theta_{k+1} = \theta_k + \alpha_k \left(r_k + \gamma \max_{a' \in \mathcal{A}} Q_{\theta_k}(h_{k+1}, a') - Q_{\theta_k}(h_k, a_k) \right) \nabla_\theta Q_{\theta_k}(h_k, a_k), \tag{2.37}$$

where $h_{k+1} = (h_k, a_k, o_{k+1})$ and where α_k is the learning rate.

Similarly, given a POMDP $\mathcal{P}$ from which we can obtain samples, the history-dependent policy-gradient approach updates a parametrized policy $\eta_\psi(a|h)$. Let us first define the discounted history-action visitation measure as follows,

$$d^{\eta,\gamma}(h, a) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \Pr(H_t = h, A_t = a). \tag{2.38}$$

Based on a transition (h_i, a_i, r_i, o_{i+1}) sampled from the discounted history-action visitation measure $d^{\eta_{\psi_i},\gamma}$ of the current policy η_{ψ_i}, the history-dependent policy-gradient approach use the following update,

$$\psi_{i+1} = \psi_i + \zeta_i Q^{\eta_{\psi_i}}(h_i, a_i) \nabla_\psi \log \eta_{\psi_i}(a_i|h_i), \tag{2.39}$$

where $Q^{\eta_{\psi_i}}$ is the Q-function of the history-dependent policy η_{ψ_i}, and ζ_i is the learning rate. In practice, the Q-function of the policy can be estimated using the empirical return, or using a history-dependent critic $Q_\theta^{\eta_{\psi_i}}$ that is updated using temporal difference learning,

$$\theta_{k+1} = \theta_k + \alpha_k \left(r_k + \gamma V_{\theta_k}^{\eta_{\psi_i}}(h_{k+1}) - Q_{\theta_k}^{\eta_{\psi_i}}(h_k, a_k) \right) \nabla_\theta Q_{\theta_k}(h_k, a_k), \tag{2.40}$$

where $V_{\theta_k}^{\eta_{\psi_i}}(h_{k+1}) = \sum_{a' \in \mathcal{A}} \eta_{\psi_i}(a'|h_{k+1}) Q_{\theta_k}^{\eta_{\psi_i}}(h_{k+1}, a')$.

Finally, for any POMDP $\mathcal{P}$, a history-dependent model of the environment can simply be learned as a model $q_\phi(r, o'|h, a)$ of the history-dependent transition function $\Pr(r, o'|h, a)$. In practice, it can be learned from samples of the environment (h_k, a_k, r_k, o_{k+1}) using any supervised learning method to minimize, for example, the following negative log likelihood loss,

$$L(h_k, a_k, r_k, o_{k+1}) = -\log q_\phi(r_k, o_{k+1}|h_k, a_k). \tag{2.41}$$

Compared to standard methods for MDPs, the only challenge is to use adequate history-dependent function approximators such as recurrent neural networks or Transformers. Usually, histories are first compressed into representations $z = f_\theta(h)$ where $f_\theta \colon \mathcal{H} \to \mathcal{Z}$ is the sequence function approximator. It

can be written $Q_\theta(h, a) = g_\theta(f_\theta(h), a)$ for the history-dependent Q-function approximator, $\eta_\psi(h|a) = g_\psi(a|f_\psi(h))$ for the history-dependent policy, and $q_\phi(r, o'|h, a) = g_\phi(r, o'|f_\phi(h), a)$ for the history-dependent world model.

As a result of this compression of the history, we have to make sure to learn a statistic $z = f_\theta(h)$ of the history h that is sufficient for predicting the Q-function, for optimal control, or for predicting the transition, respectively. This is the concern of this thesis, which will notably investigate the representations that are learned by these methods, develop methods to learn sufficient representations, and study the impact of insufficient representations.

Part I

Learning and Remembering

Learning and Remembering

In this part, we question the interplay between memory and learning. First, we study the internal representations that are learn by recurrent model-free reinforcement learning in partially observable Markov decision processes. We empirically verify that the hidden states of recurrent reinforcement learning agent naturally encodes the posterior distribution over the states, known as the belief. Second, we study a new initialization procedure for recurrent neural networks that endows them with a long lasting memory. More precisely, we empirically show that maximizing their number of attractors results in a better memory and a better learning ability.

Chapter 3

Learning for Remembering

Recurrent Networks, Hidden States and Beliefs in Partially Observable Environments. Gaspard Lambrechts, Adrien Bolland and Damien Ernst.

From the paper published in the *Transactions on Machine Learning Research*.

Abstract

Reinforcement learning aims to learn optimal policies from interaction with environments whose dynamics are unknown. Many methods rely on the approximation of a value function to derive near-optimal policies. In partially observable environments, these functions depend on the complete sequence of observations and past actions, called the history. In this work, we empirically show that recurrent neural networks trained to approximate such value functions internally filter the posterior probability distribution of the current state given the history, called the belief. More precisely, we show that, as a recurrent neural network learns the Q-function, its hidden states become more and more correlated with the beliefs of state variables that are relevant to optimal control. This correlation is measured through their mutual information. In addition, we show that the expected return of an agent increases with the ability of its recurrent architecture to reach a high mutual information between its hidden states and the beliefs. Finally, we show that the mutual information between the hidden states and the beliefs of variables that are irrelevant for optimal control decreases through the learning process. In summary, this work shows that in its hidden states, a recurrent neural network approximating the Q-function of a partially observable environment learns a sufficient statistic of the history that is correlated to the relevant part of the belief for taking optimal actions.

3.1 Introduction

Latest advances in reinforcement learning (RL) rely heavily on the ability to approximate a value function (i.e., state or state-action value function). Modern RL algorithms have been shown to be able to produce approximations of the value functions of Markov decision processes (MDP) from which high-quality policies can be derived, even in the case of continuous and high-dimensional state and action spaces [Mnih et al., 2015, Lillicrap et al., 2015, Mnih et al., 2016, Haarnoja et al., 2018, Hessel et al., 2018]. The adaptation of these techniques to partially observable MDPs (POMDP) is not straightforward. Indeed, in such environments, the agent only receives partial observations of the underlying state of the environment. Unlike MDPs where the value functions are written as functions of the current state, in POMDPs the value functions are written as functions of the complete sequence of observations and past actions, called the history. Moreover, the value functions of a history can equivalently be written as functions of the posterior probability distribution over the current state given this history [Åström, 1965]. This posterior probability distribution is called the belief and is said to be a sufficient statistic of the history for the value functions of the POMDP [Striebel, 1965]. However, the computation of the belief requires the POMDP model to be known and is generally intractable with large or continuous state spaces. For these two reasons, practical RL algorithms rely on the definition of the value functions as functions of the complete history (i.e., history or history-action value function), while the definition of the value functions as functions of the belief (i.e., belief or belief-action value function) is more of theoretical interest.

Approximating the value functions as functions of the histories requires one to use function approximators that are able to process sequences of arbitrary length. In practice, RNNs are good candidates for such approximators [Bakker, 2001, Wierstra et al., 2007, Hausknecht and Stone, 2015, Heess et al., 2015]. RNNs are parametric approximators that process sequences, time step by time step, exhibiting memory through a hidden state that is passed recurrently over time. The RNN is thus tasked with outputting the value directly from the history. We focus on the approximation of the history-action value function, or Q-function, using a parametric recurrent Q-learning (PRQL) algorithm. More precisely, RNNs are trained with the deep recurrent Q-network (DRQN) algorithm [Hausknecht and Stone, 2015, Zhu et al., 2017].

Since we know that the belief is a sufficient statistic of the history for the Q-function of this history [Åström, 1965], we investigate whether RNNs, once trained, reproduce the belief filter when processing a history. This investigation is conducted in this work by studying the performance of the different agents with regard to the mutual information (MI) between their hidden states and the belief. We focus on POMDPs for which the models are known. The benchmarks chosen are the T-Maze environments [Bakker, 2001] and the Mountain Hike environments [Igl et al., 2018]. The first ones present a discrete state space, allowing one to compute the belief using Bayes' rule, and representing this distribution over the states in a vector whose dimension is equal to the number of states. The second ones present a continuous state space, making the belief update intractable. We thus rely on particle filtering in order to approximate the belief by a set of states, called particles, distributed according to the belief

distribution. The MI between the hidden states and the beliefs is periodically estimated during training, using the mutual information neural estimator (MINE) algorithm [Belghazi et al., 2018]. The MINE estimator is extended with the deep set architecture [Zaheer et al., 2017] in order to process sets of particles in the case of POMDPs with continuous state-spaces. This methodology allows one to measure the ability and tendency of recurrent architecture to reproduce the belief filter when trained to approximate the Q-function.

In [Mikulik et al., 2020], a similar study is performed in the meta-learning setting. In this setting, an MDP is drawn from a distribution of MDPs at each episode. This problem can be equivalently modeled as a particular subclass of POMDP [Ghosh et al., 2021]. The authors show empirically, among others, that the hidden state of an RNN-based policy and the statistic of the optimal policy can be mapped one into the other with a low dissimilarity measure. In contrast, we consider arbitrary POMDPs and show empirically that information about the belief, a statistic known to be sufficient for optimal control, is encoded in the hidden states.

In Section 3.2, we formalize the problem of optimal control in POMDPs, we present the PRQL algorithm for deriving near-optimal policies and we explain the MINE algorithm for estimating the MI. In Section 3.3, the beliefs and hidden states are defined as random variables whose MI is measured. Section 3.4 gives the results obtained for the considered POMDPs. Finally, Section 3.5 concludes and proposes several future works and algorithms motivated by our results.

3.2 Background

In Subsection 3.2.1, POMDPs are introduced, along with the belief, policy, and Q-functions associated with such decision processes. In Subsection 3.2.2, we introduce the DRQN algorithm that is used in our experiments. This algorithm is a particular instance of the PRQL class of algorithms that allows to approximate the Q-function for deriving a near-optimal policy in a POMDP. Finally, in Subsection 3.2.3, we present the MINE algorithm that is used for estimating the MI between the hidden states and beliefs in our experiments.

3.2.1 Partially Observable Markov Decision Processes

In this work, the environments are modeled as POMDPs. Formally, a POMDP $\mathcal{P}$ is a tuple $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, P, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, and $\mathcal{O}$ is the observation space. The initial state distribution P gives the probability $P(s_0)$ of $s_0 \in \mathcal{S}$ being the initial state of the decision process. The dynamics are described by the transition distribution T that gives the probability $T(s_{t+1}|s_t, a_t)$ of $s_{t+1} \in \mathcal{S}$ being the state resulting from action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$. The reward function R gives the immediate reward $r_t = R(s_t, a_t, s_{t+1})$ obtained after each transition. The observation distribution O gives the probability $O(o_t|s_t)$ to get observation $o_t \in \mathcal{O}$ in state $s_t \in \mathcal{S}$. Finally, the discount factor $\gamma \in [0, 1)$ weights the relative importance of future rewards.

Taking a sequence of t actions $(a_{0:t-1})$ in the POMDP conditions its execution and provides a sequence of $t + 1$ observations $(o_{0:t})$. Together, they compose

the history $h_t = (o_0, a_0, \ldots, o_t) \in \mathcal{H}_t$ until time step t, where $\mathcal{H}_t$ is the set of such histories. Let $h \in \mathcal{H}$ denote a history of arbitrary length sampled in the POMDP, with $\mathcal{H} = \bigcup_{t=0}^{\infty} \mathcal{H}_t$ the set of histories of arbitrary length.

In the following, we denote with uppercase letters the random variables S_t, A_t, O_t, H_t, R_t, and with lowercase letters their realizations s_t, a_t o_t, h_t and r_t.

A policy $\eta \in H$ in a POMDP is a mapping from histories to actions, where $H = \mathcal{H} \to \mathcal{A}$ is the set of such mappings. A policy $\eta^* \in H$ is said to be optimal when it maximizes the expected discounted sum of future rewards starting from any history $h \in \mathcal{H}$,

$$\eta^* \in \arg\max_{\eta \in H} \mathbb{E}^{\eta} \left[\sum_{t=0}^{\infty} \gamma^t R_t \;\middle|\; H_0 = h \right], \; \forall h \in \mathcal{H}. \tag{3.1}$$

The history-action value function, or Q-function, is defined as the maximal expected discounted reward that can be gathered, starting from a history $h \in \mathcal{H}$ and an action $a \in \mathcal{A}$,

$$Q(h, a) = \max_{\eta \in H} \mathbb{E}^{\eta} \left[\sum_{t=0}^{\infty} \gamma^t R_t \;\middle|\; H_0 = h, A_0 = a \right], \tag{3.2}$$

The Q-function is also the unique solution of the Bellman equation [Smallwood and Sondik, 1973, Kaelbling et al., 1998, Porta et al., 2006],

$$Q(h, a) = \mathbb{E}\left[R + \gamma \max_{a' \in \mathcal{A}} Q(H', A') \;\middle|\; H = h, A = a \right], \tag{3.3}$$

where $H' = (H, A, O')$ and R is the immediate reward obtained when taking action A in history H. From equation (3.1) and equation (3.2), it can be observed that any optimal policy satisfies,

$$\eta^*(h) \in \arg\max_{a \in \mathcal{A}} Q(h, a), \; \forall h \in \mathcal{H}. \tag{3.4}$$

Let $\Delta(\mathcal{S})$ be the set of probability measures over the state space $\mathcal{S}$. The belief $b \in \Delta(\mathcal{S})$ of a history $h \in \mathcal{H}$ is defined as the posterior probability distribution over the states given the history, such that $b(s) = p(s|h), \; \forall s \in \mathcal{S}$ [Thrun, 2002]. The belief filter f is defined as the function that maps a history h to its corresponding belief b,

$$f(h) = b, \; \forall h \in \mathcal{H}. \tag{3.5}$$

Formally, for an initial observation $h = (o)$, the belief $b = f(h)$ is defined by,

$$b(s) = \frac{P(s) O(o|s)}{\int_{\mathcal{S}} P(s') O(o|s') \, \mathrm{d}s'}, \; \forall s \in \mathcal{S} \tag{3.6}$$

and for a history $h' = (h, a, o')$, the belief $b' = f(h')$ is recursively defined by,

$$b'(s') = \frac{O(o'|s') \int_{\mathcal{S}} T(s'|s, a) \; b(s) \, \mathrm{d}s}{\int_{\mathcal{S}} O(o'|s') \int_{\mathcal{S}} T(s'|s, a) \; b(s) \, \mathrm{d}s \, \mathrm{d}s'}, \; \forall s' \in \mathcal{S}. \tag{3.7}$$

where $b = f(h)$. Equation (3.7) provides a way to update the belief b to b' through a filter step f once observing new information (a, o'),

$$b' = u(b; a, o'). \tag{3.8}$$

A statistic of the history is defined as any function of the history. The belief is known to be a sufficient statistic of the history in order to act optimally [Bertsekas, 2012]. It means that the Q-function only depends on the history through the belief computed from this same history. It implies in particular that the Q-function takes the following form,

$$Q(h, a) = Q'(f(h), a), \ \forall h \in \mathcal{H}, \ \forall a \in \mathcal{A} \tag{3.9}$$

where $Q' \colon \Delta(\mathcal{S}) \times \mathcal{A} \to \mathbb{R}$ is called the belief-action value function, or belief Q-function. This function gives the maximal expected discounted reward starting from a belief $b \in \Delta(\mathcal{S})$ and an action $a \in \mathcal{A}$, where the belief $b = f(h)$ results from an arbitrary history $h \in \mathcal{H}$. Although the exact belief filter is often unknown or intractable, this factorization of the Q-function still motivates the compression of the history in a statistic related to the belief, when processing the history for predicting the Q-function.

3.2.2 Parametric Recurrent Q-learning

We call PRQL the family of algorithms that aim at learning an approximation of the Q-function with a recurrent architecture Q_θ, where $\theta \in \mathbb{R}^{d_\theta}$ is the parameter vector. These algorithms are motivated by equation (3.4) that shows that an optimal policy can be derived from the Q-function. The strategy consists of minimising, with respect to θ, for all (h, a), the distance between the estimation $Q_\theta(h, a)$ of the LHS of equation (3.3), and the estimation of the expectation $\mathbb{E}[r + \gamma \max_{a' \in \mathcal{A}} Q_\theta(h', a')]$ of the RHS of equation (3.3). This is done by using transitions (h, a, r, o', h') sampled in the POMDP, with $h' = (h, a, o')$. In its simplest form, given such a transition, the PRQL algorithm updates the parameters $\theta \in \mathbb{R}^{d_\theta}$ of the function approximator according to,

$$\theta \leftarrow \theta + \alpha \left(r + \gamma \max_{a' \in \mathcal{A}} \{Q_\theta(h', a')\} - Q_\theta(h, a) \right) \nabla_\theta Q_\theta(h, a). \tag{3.10}$$

This update corresponds to a gradient step in the direction that minimizes, with respect to θ the squared distance between $Q_\theta(h, a)$ and the target $r + \gamma \max_{a' \in \mathcal{A}} \{Q_\theta(h', a')\}$ considered independent of θ. It can be noted that, in practice, such algorithms introduce a truncation horizon H such that the histories generated in the POMDP have a maximum length of H. From the approximation Q_θ, the policy η_θ is given by $\eta_\theta(h) = \arg\max_{a \in \mathcal{A}} Q_\theta(h, a)$. Equation (3.4) guarantees the optimality of this policy if $Q_\theta = Q$. Even though it will alter the performance of the algorithm, any policy can be used to sample the transitions (h, a, r, o', h').

The function approximator Q_θ of PRQL algorithms should be able to process inputs $h \in \mathcal{H}$ of arbitrary length, making RNN approximators a suitable choice. Indeed, RNNs process the inputs sequentially, exhibiting memory through hidden states that are outputted after each time step, and processed at the next

time step along with the following input. More formally, let $x_{0:t} = [x_0, \dots, x_t]$ with $t \in \mathbb{N}_0$ be an input sequence. At any step $k \in \{0, \dots, t\}$, RNNs maintain an internal memory state z_k through the update function (3.11) and output a value y_k through the output function (3.12). The initial state z_{-1} is given by the initialization function (3.13).

$$z_k = u_\theta(z_{k-1}, x_k),\ \forall k \in \mathbb{N}_0, \tag{3.11}$$

$$y_k = o_\theta(z_k),\ \forall k \in \mathbb{N}_0, \tag{3.12}$$

$$z_{-1} = i_\theta. \tag{3.13}$$

These networks are trained based on backpropagation through time where gradients are computed in a backward pass through the complete sequence via the hidden states [Werbos, 1990]. The following recurrent architectures are used in the experiments: the long short-term memory (LSTM) by Hochreiter and Schmidhuber [1997], the gated recurrent unit (GRU) by Chung et al. [2014], the bistable recurrent cell (BRC) and recurrently neuromodulated bistable recurrent cell (nBRC) by Vecoven et al. [2021], and the minimal gated unit (MGU) by Zhou et al. [2016].

In the experiments, we use the DRQN algorithm [Hausknecht and Stone, 2015, Zhu et al., 2017] to learn policies. This algorithm is a PRQL algorithm that shows good convergence even for high-dimensional problems. The DRQN algorithm is detailed in Algorithm 3.1 of Appendix 3.B. In this algorithm, for a given history h_t of arbitrary length t, the inputs of the RNN are $x_k = (a_{k-1}, o_k),\ k = 1, \dots, t$ and $x_0 = (0, o_0)$, and the output of the RNN at the last time step $y_t = o_\theta(z_t) \in \mathbb{R}^{|\mathcal{A}|}$ gives $y_t^{a_t} = Q_\theta(h_t, a_t)$, for any $a_t \in \mathcal{A}$. We also define the composition $f_\theta \colon \mathcal{H} \to \mathbb{R}^{d_z}$ of equation (3.13) and equation (3.11) applied on the complete history, such that,

$$z_t = f_\theta(h_t) = \begin{cases} u_\theta(f_\theta(h_{t-1}), x_t), & t \geq 1 \\ u_\theta(i_\theta, x_t), & t = 0 \end{cases} \tag{3.14}$$

3.2.3 Mutual Information Neural Estimator

In this work, we are interested in establishing whether a recurrent function approximator reproduces the belief filter during PRQL. Formally, this is performed by estimating the MI between the beliefs and the hidden states of the RNN approximator Q_θ. In this subsection, we recall the concept of MI and how it can be estimated in practice.

The MI is theoretically able to measure any kind of dependency between random variables [Kraskov et al., 2004]. The MI between two jointly continuous random variables X and Y is defined as,

$$I(X;Y) = \int_{\mathcal{X}} \int_{\mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p_X(x)\, p_Y(y)} \,\mathrm{d}x \,\mathrm{d}y \tag{3.15}$$

where $\mathcal{X}$ and $\mathcal{Y}$ are the support of the random variables X and Y respectively, p is the joint probability density function of X and Y, and p_X and p_Y are the marginal probability density functions of X and Y, respectively. It is worth noting that the MI can be defined in terms of the Kulback-Leibler (KL) divergence between the joint p and the product of the marginals $q = p_X \otimes p_Y$, over

the joint space $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$,

$$I(X;Y) = D_{\mathrm{KL}}(p \parallel q) = \int_{\mathcal{Z}} p(z) \log \left(\frac{p(z)}{q(z)} \right) \mathrm{d}z \tag{3.16}$$

In order to estimate the MI between random variables X and Y from a dataset $\{(x_i, y_i)\}_{i=1}^{N}$, we rely on the MINE algorithm [Belghazi et al., 2018]. This technique is a parametric approach where a neural network outputs a lower bound on the MI, that is maximized by gradient ascent. The lower bound is derived from the Donsker-Varhadan representation of the KL-divergence [Donsker and Varadhan, 1975],

$$D_{\mathrm{KL}}(p \parallel q) = \sup_{T\colon \mathcal{Z} \to \mathbb{R}} \mathbb{E}_{z \sim p}\left[T(z)\right] - \log \left(\mathbb{E}_{z \sim q}\left[e^{T(z)} \right] \right) \tag{3.17}$$

where the supremum is taken over all functions T such that the two expectations are finite. The lower bound $I_\Phi(X;Y)$ on the true MI $I(X;Y)$ is obtained by replacing T by a parameterized function $T_\phi\colon \mathcal{Z} \to \mathbb{R}$ with $\phi \in \Phi$, and taking the supremum over the parameter space Φ of this function. If Φ corresponds to the parameter space of a neural network, then this lower bound can be approached by gradient ascent using empirical means as estimators of the expectations. The resulting procedure for estimating the MI is given in Algorithm 3.3 in Appendix 3.D.

3.3 Measuring the Correlation Between Hidden States and Beliefs

In this work, we study if PRQL implicitly approximates the belief filter by reaching a high MI between the RNN's hidden states and the beliefs, that are both generated from random histories. In this section, we first explain the intuition behind this hypothesis, then we define the joint probability distribution over the hidden states and beliefs that defines the MI.

As explained in Section 3.2, the belief filter is generally intractable. As a consequence, PRQL algorithms use approximators Q_θ that directly take the histories as input. In the DRQN algorithm, these histories are processed recurrently according to equation (3.11), producing a new hidden state z_t after each input $x_t = (a_{t-1}, o_t)$,

$$z_t = u_\theta(z_{t-1}; (a_{t-1}, o_t)). \tag{3.18}$$

These hidden states should thus summarize all relevant information from past inputs in order to predict the Q-function at all later time steps. The belief is known to be a sufficient statistic of the history for these predictions (3.9). Moreover, the belief b_t is also updated recurrently, according to equation (3.8) after each transition (a_{t-1}, o_t),

$$b_t = u(b_{t-1}; a_{t-1}, o_t). \tag{3.19}$$

The parallel between equation (3.18) and equation (3.19), knowing the sufficiency of the belief (3.9), justifies the appropriateness of the belief update u as the update function u_θ of the RNN approximator Q_θ. It motivates the study of the reconstruction of the belief filter by the RNN.

This is done through the estimation of the MI between the hidden state z_t and the belief b_t at any time step $t \in \mathbb{N}_0$. Formally, for a given history length $t \in \mathbb{N}_0$, the policy η_θ of the learning algorithm, as defined in Subsection 3.2.2, induces a distribution $p_{\eta_\theta}(h|t)$ over histories $h \in \mathcal{H}$. This conditional probability distribution is zero for all history of length $t' \neq t$. Given a distribution $p(t)$ over trajectory lengths, the joint distribution $p(z, b)$ is given by,

$$p(z, b) = \sum_{t=0}^{\infty} p(t) \int_{\mathcal{H}} p(z, b|h)\, p_{\eta_\theta}(h|t)\, \mathrm{d}h \tag{3.20}$$

where $p(z, b|h)$ is a Dirac distribution for $z = f_\theta(h)$ and $b = f(h)$ given by equation (3.14) and equation (3.5), respectively. In the following, we estimate the MI between z and b under their joint distribution (3.20).

3.4 Experiments

In this section, the experimental protocol and environments are described and the results are given. More specifically, in Subsection 3.4.1, we describe the estimates that are reported in the figures. The results are reported for four different POMDPs: the T-Maze and Stochastic T-Maze in Subsection 3.4.2, and the Mountain Hike and Varying Mountain Hike in Subsection 3.4.3. Afterwards, in Subsection 3.4.4, irrelevant state variables and observations are added to the decision processes, and the MI is measured separately between the hidden states and the belief of the relevant and irrelevant variables. Finally, in Subsection 3.4.5, we discuss the results obtained in this section, and propose an additional protocol to study their generalization.

3.4.1 Experimental Protocol

As explained in Subsection 3.2.2, the parameters θ of the approximation Q_θ are optimized with the DRQN algorithm. After e episodes of interaction with the POMDP, the DRQN algorithm gives the policy $\eta_{\theta_e}(h) = \arg\max_{a \in \mathcal{A}} Q_{\theta_e}(h, a)$. In the experiments, the empirical return $\hat{J}(\theta_e)$ of the policy η_{θ_e} is reported, along with the estimated MI $\hat{I}(\theta_e)$ between the random variables z and b under the distribution (3.20) implied by η_{θ_e}. Each estimate is reported averaged over four training sessions. In addition, confidence intervals show the minimum and maximum of these estimates.

The empirical return is defined as $\hat{J}(\theta_e) = \frac{1}{I} \sum_{i=0}^{I-1} \sum_{t=0}^{H-1} \gamma^t r_t^i$, where I is the number of Monte Carlo rollouts, H the truncation horizon of the DRQN algorithm, and r_t^i is the reward obtained at time step t of Monte Carlo rollout i. As far as the estimation of the MI is concerned, we sample time steps with equal probability $p(t) = 1/H,\ t \in \{0, \dots, H-1\}$, where H is the truncation horizon of the DRQN algorithm. The uniform distribution over time steps and the current policy η_{θ_e} define the probability distribution (3.20) over the hidden states and beliefs. The MI is estimated from samples of this distribution using the MINE estimator $\hat{I}(\theta_e)$ (see Subappendix 3.D.1 for details). The hyperparameters of the DRQN and MINE algorithms are given in Appendix 3.E.

For POMDPs with continuous state spaces, the computation of the belief b is intractable. However, a set of state particles S that follows the belief distribution

$f(h)$ can be sampled, using particle filtering (see Appendix 3.C). This set of particles could be used to construct an approximation of the belief in order to estimate the MI. This density estimation procedure is nonetheless unnecessary as the MINE network can directly process the set of particles by producing a permutation-invariant embedding of the belief using the deep set architecture [Zaheer et al., 2017], see Subappendix 3.D.2 for details.

3.4.2 Deterministic and Stochastic T-Mazes

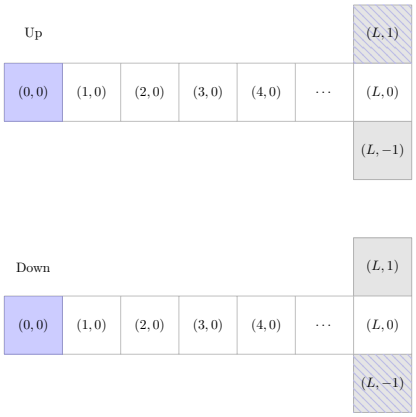


Figure 3.1: T-Maze state space.

The T-Maze is a POMDP where the agent is tasked with finding the treasure in a T-shaped maze (see Figure 3.1). The state is given by the position of the agent in the maze and the maze layout that indicates whether the treasure lies up or down after the crossroads. The initial state determines the maze layout, and it never changes afterwards. The initial observation made by the agent indicates the layout. Navigating in the maze provides zero reward, except when bouncing onto a wall, in which case a reward of -0.1 is received. Finding the treasure provides a reward of 4. Beyond the crossroads, the states are always terminal. The optimal policy thus consists of going through the maze, while remembering the initial observation in order to take the correct direction at the crossroads. This POMDP is parameterized by the corridor length $L \in \mathbb{N}$ and stochasticity rate $\lambda \in [0, 1]$ that gives the probability of moving in a random direction at any time step. The Deterministic T-Maze ($\lambda = 0$) was originally proposed in [Bakker, 2001]. The discount factor is $\gamma = 0.98$. This POMDP is formally defined in Subappendix 3.A.2.

As explained in Subsection 3.2.2, the histories can be sampled with an arbitrary policy in PRQL algorithms. In practice, the DRQN algorithm uses an ε-greedy stochastic policy that selects its action according to the current policy with probability $1-\varepsilon$, and according to the exploration policy $\mathcal{E}(\mathcal{A})$ with probability ε. Usually, the exploration policy is chosen to be the uniform distribution $\mathcal{U}(\mathcal{A})$ over the action. However, for the T-Maze, the exploration policy $\mathcal{E}(\mathcal{A})$ is tailored to this POMDP to alleviate the exploration problem, that is independent of the study of this work. The exploration policy forces one to walk through the

right of the corridor with $\mathcal{E}(\text{Right}) = 1/2$ and $\mathcal{E}(\text{Other}) = 1/6$ where Other $\in \{\text{Up}, \text{Left}, \text{Down}\}$.

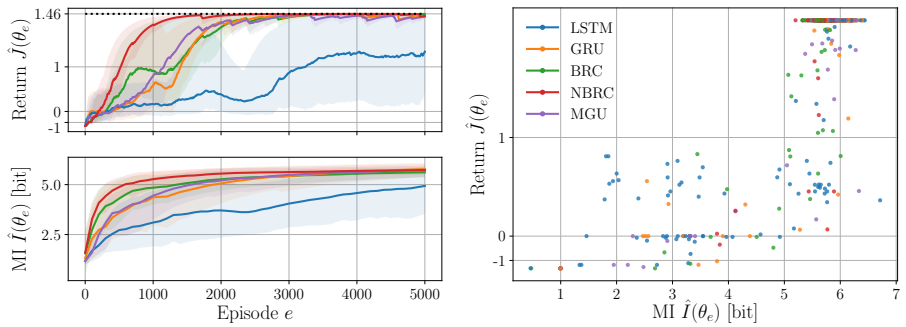


Figure 3.2: Deterministic T-Maze ($L = 50$). Evolution of the return $\hat{J}(\theta_e)$ and the mutual information $\hat{I}(\theta_e)$ after e episodes (left), and the return $\hat{J}(\theta_e)$ with respect to the mutual information $\hat{I}(\theta_e)$ (right). The maximal expected return is given by the dotted line.

On the left in Figure 3.2, the expected return is shown along with the MI between the hidden states and the belief as a function of the number of episodes, for a T-Maze of length $L = 50$. In order to better disambiguate between high-quality policies, the empirical return is displayed with an exponential scale in the following graphs. Both the performance of the policy and the MI increase during training. We also observe that, at any given episode, RNNs that have a higher return, such as the nBRC or the BRC, correspond to cells that have a higher MI between their hidden states and the belief. Furthermore, the LSTM that struggles to achieve a high return has a significantly lower MI than the other cells. Finally, we can see that the evolution of the MI and the return are correlated, which is highlighted on the right in Figure 3.2. Indeed, the return increases with the MI, with a linear correlation coefficient of 0.8233 and a rank correlation coefficient of 0.6419. These correlations coefficients are also detailed for each cell separately in Appendix 3.G. It can also be noted that no RNN with less than 5 bits of MI reaches the maximal return.

In Figure 3.3, we can see that all previous observations also hold for a T-Maze of length $L = 100$. On the left, we can see that the lower the MI, the lower the return of the policy. For this length, in addition to the LSTM, the GRU struggles to achieve the maximal return, which is reflected in the evolution of its MI that increases more slowly than for the other RNNs. It is also interesting to notice that, on average, the MGU overtake the BRC in term of return after 2000 episodes, which is also the case for the MI. Here, the linear correlation coefficient between the MI and the return is 0.5347 and the rank correlation coefficient is 0.6666. Once again, we observe that a minimum amount of MI between the hidden states and the belief is required for the policy to be optimal. Here, at least 5.0 bits of MI is necessary.

In Figure 3.4, the results are shown for the Stochastic T-Maze with $L = 50$ and $\lambda = 0.3$. On the contrary to the Deterministic T-Maze, where the belief is a Dirac distribution over the states, there is uncertainty on the true state

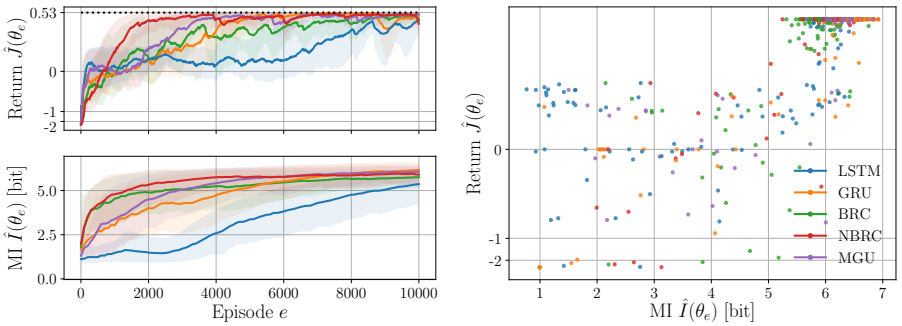


Figure 3.3: Deterministic T-Maze ($L = 100$). Evolution of the return $\hat{J}(\theta_e)$ and the mutual information $\hat{I}(\theta_e)$ after e episodes (left), and the return $\hat{J}(\theta_e)$ with respect to the mutual information $\hat{I}(\theta_e)$ (right). The maximal expected return is given by the dotted line.

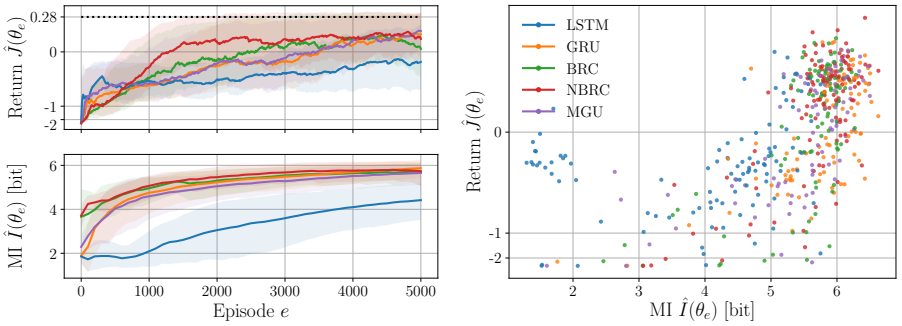


Figure 3.4: Stochastic T-Maze ($L = 50$, $\lambda = 0.3$). Evolution of the return $\hat{J}(\theta_e)$ and the mutual information $\hat{I}(\theta_e)$ after e episodes (left), and the return $\hat{J}(\theta_e)$ with respect to the mutual information $\hat{I}(\theta_e)$ (right). The maximal expected return is given by the dotted line.

in this environment. We can nevertheless observe that previous observations hold for this environment too. The MI and the expected return are indeed both increasing throughout the training process, and the best performing RNNs, such as the BRC and nBRC, have a MI that increases faster and stays higher, while the LSTM struggles to reach both a high return and a high MI. Here, the linear correlation coefficient between the MI and the return is 0.5460 and the rank correlation coefficient is 0.6403. It can also be noticed on the right that the best performing policies have a MI of at least 4.5 bits in practice.

In the Deterministic T-Maze, it can be observed that the estimated lower bounds $I_\phi(z, b)$ on the MI that are obtained by the MINE estimator are tight. Indeed, in this environment, the hidden state and belief are discrete random variables and their mutual information is thus upper bounded by the entropy of the belief. Moreover, the belief is a Dirac distribution that gives the actual state with probability one. Under the optimal policy, each state is visited with equal probability, such that the entropy of the belief is given by $\log_2(102) = 6.6724$

for the Deterministic T-Maze of length $L = 50$, where 102 is the number of non terminal states. As can be seen in Figure 3.2, the optimal policies reach an estimated MI around 6.5 at maximum, which nearly equals the upper bound. The same results is obtained for the Deterministic T-Maze of length $L = 100$, where the entropy of the belief is given by $\log_2(202) = 7.658$ and the optimal policies reach an estimated MI around 7.0 at maximum, as can be seen in Figure 3.3. We expect this result to generalize to other environments even if this would be difficult to verify in practice for random variables with large or continuous spaces.

3.4.3 Mountain Hike and Varying Mountain Hike

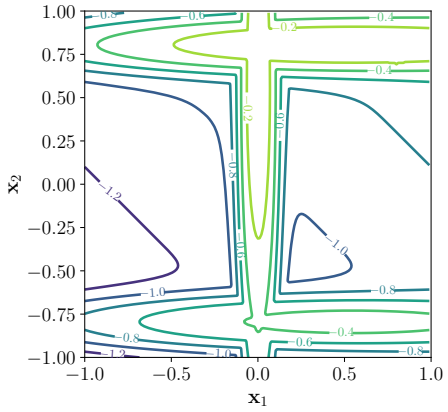


Figure 3.5: Mountain Hike altitude function.

The Mountain Hike environment is a POMDP modeling an agent walking through a mountainous terrain. The agent has a position on a two-dimensional map and can take actions to move in four directions relative to its initial orientation: Forward, Backward, Right and Left. First, we consider that its initial orientation is always North. Taking an action results in a noisy translation in the corresponding direction. The translation noise is Gaussian with a standard deviation of $\sigma_T = 0.05$. The only observation available is a noisy measure of its relative altitude to the mountain top, that is always negative. The observation noise is Gaussian with a standard deviation of $\sigma_O = 0.1$. The reward is also given by this relative altitude, such that the goal of this POMDP is to to obtain the highest possible cumulative altitude. Around the mountain top, the states are terminal. The optimal policy thus consists of going as fast as possible towards those terminal states while staying on the crests in order to get less negative rewards than in the valleys. This environment is represented in Figure 3.5. This POMDP is inspired by the Mountain Hike environment described in [Igl et al., 2018]. The discount factor is $\gamma = 0.99$. We also consider the Varying Mountain Hike in the experiments, a more difficult version of the Mountain Hike where the agent randomly faces one of the four cardinal directions (i.e., North, West, South, East) depending on the initial state. The agent does not observe its orientation. As a consequence, the agent needs to maintain a belief about its orientation given the observations in order to act optimally. This POMDP is

formally defined in Subappendix 3.A.3.

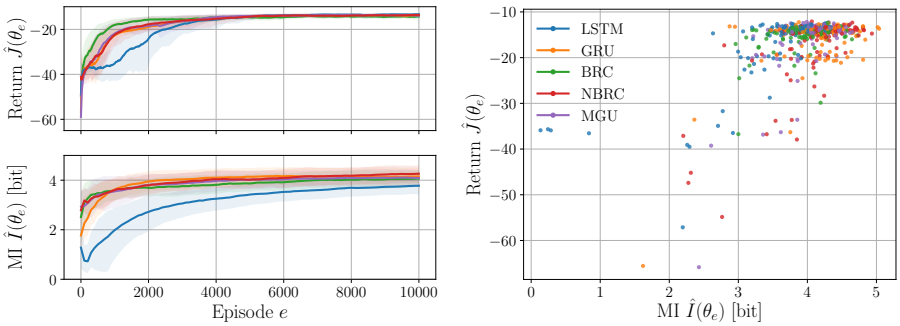


Figure 3.6: Mountain Hike. Evolution of the return $\hat{J}(\theta_e)$ and the mutual information $\hat{I}(\theta_e)$ after e episodes (left), and the return $\hat{J}(\theta_e)$ with respect to the mutual information $\hat{I}(\theta_e)$ (right).

Figure 3.6 shows on the left the expected return and the MI during training for the Mountain Hike environment. It is clear that the DRQN algorithm promotes a high MI between the belief and the hidden states of the RNN, even in continuous-state environments. It can also be seen that the evolution of the MI and the evolution of the return are strongly linked throughout the training process, for all RNNs. We can also see on the right in Figure 3.6 that the correlation between MI and performances appears clearly for each RNN. For all RNNs, the linear correlation coefficient is 0.5948 and the rank correlation coefficient is 0.2965. In particular, we see that the best policies, with a return around -20, are clearly separated from the others and have a significantly higher MI on average.

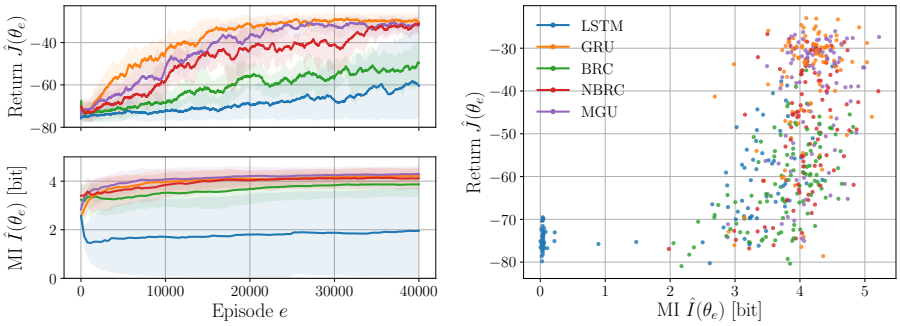


Figure 3.7: Varying Mountain Hike. Evolution of the return $\hat{J}(\theta_e)$ and the mutual information $\hat{I}(\theta_e)$ after e episodes (left), and the return $\hat{J}(\theta_e)$ with respect to the mutual information $\hat{I}(\theta_e)$ (right).

In Figure 3.7, we can see the evolution and the correlation between the return and the MI for the Varying Mountain Hike environment. The correlation is even clearer than for the other environments. This may be due to the fact that differences in term of performances are more pronounced than for the other experiments. Again, the worse RNNs such as the LSTM and the BRC have a

significantly lower MI compared to the other cells. In addition, the performances of any RNN is strongly correlated to their ability to reproduce the belief filter, as can be seen on the right, with a sharp increase in empirical return as the MI increases from 2.5 to 4.5 bits. More precisely, the linear correlation coefficient between the MI and the return is 0.5982 and the rank correlation coefficient is 0.6176. This increase occurs throughout the training process, as can be seen on the left.

3.4.4 Belief of Variables Irrelevant for Optimal Control

Despite the belief being a sufficient statistic of the history in order to act optimally, it may be that only the belief of some state variables is necessary for optimal control. In this subsection, we show that approximating the Q-function with an RNN will only tend to reconstruct the necessary part, naturally filtering away the belief of irrelevant state variables.

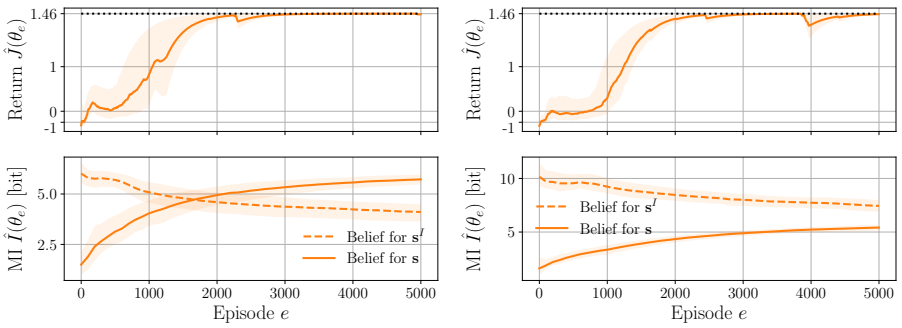
In order to study this phenomenon, we construct a new POMDP $\mathcal{P}'$ from a POMDP $\mathcal{P}$ by adding new state variables, independent of the original ones, and irrelevant for optimal control. More precisely, we add d irrelevant state variables s_t^I that follows a Gaussian random walk. In addition, the agent acting in the POMDP $\mathcal{P}'$ obtains partial observations o_t^I of the new state variables through an unbiased Gaussian observation model. Formally, the new states and observations are distributed according to,

$$p(s_0^I) = \phi(s_0^I; 0, 1), \tag{3.21}$$

$$p(s_{t+1}^I | s_t^I) = \phi(s_{t+1}^I; s_t^I, 1), \ \forall t \in \mathbb{N}_0, \tag{3.22}$$

$$p(o_t^I | s_t^I) = \phi(o_t^I; s_t^I, 1), \ \forall t \in \mathbb{N}_0, \tag{3.23}$$

where $\phi(x; \mu, \Sigma)$ is the probability density function of a multivariate random variable of mean $\mu \in \mathbb{R}^d$ and covariance matrix $\Sigma \in \mathbb{R}^{d \times d}$, evaluated at $x \in \mathbb{R}^d$, and 1 is the identity matrix.

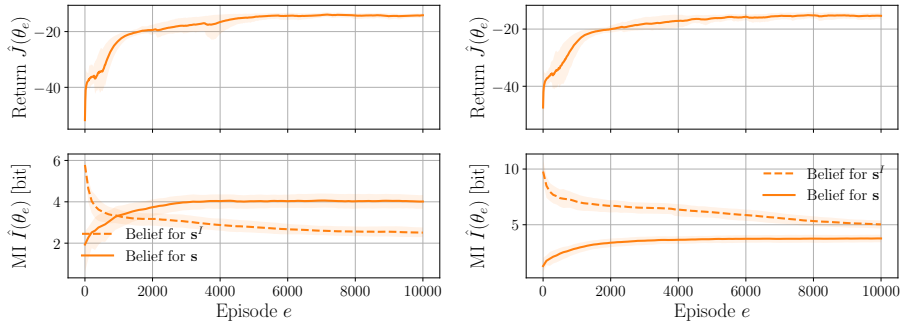


(a) $d = 1$ (b) $d = 4$

Figure 3.8: Deterministic T-Maze ($L = 50$) with d irrelevant state variables. Evolution of the return $\hat{J}(\theta_e)$ and the mutual information $\hat{I}(\theta_e)$ for the belief of the irrelevant and relevant state variables after e episodes, for the GRU cell. The maximal expected return is given by the dotted line.

Figure 3.8 shows the return and the MI measured for the GRU on the T-Maze environment with $L = 50$. It can be observed, as for the classic T-Maze envi-

ronment, that the MI between the hidden states and the belief of state variables that are relevant to optimal control increases with the return. In addition, the MI with the belief of irrelevant variables decreases during training. It can also be seen that, for $d = 4$, the MI with the belief of irrelevant variables remains higher than the MI with the belief of relevant variables, due to the high entropy of this irrelevant process. Finally, it is interesting to note that the MI continues to increase (resp. decrease) with the belief of relevant (resp. irrelevant) variables long after the optimal policy is reached, suggesting that the hidden states of the RNN still change substantially. Similar results are obtained for the other cells (see Appendix 3.H).



(a) $d = 1$ (b) $d = 4$

Figure 3.9: Mountain Hike with d irrelevant state variables. Evolution of the return $\hat{J}(\theta_e)$ and the mutual information $\hat{I}(\theta_e)$ for the belief of the irrelevant and relevant state variables after e episodes, for the GRU cell.

Figure 3.9 shows the return and the MI measured for the GRU on the Mountain Hike environment. The same conclusions as for the T-Maze can be drawn, with a clear increase of the MI for the relevant variables throughout the training process, and a clear decrease of the MI for the irrelevant variables. In addition, it can be seen that the optimal policy is reached later when there are more irrelevant variables. It is also clear that adding more irrelevant variables increases the entropy of the irrelevant process, which leads to a higher MI between the hidden states and the irrelevant state variables. Similar results are obtained for the other cells (see Appendix 3.H).

3.4.5 Discussion

As shown in the experiments, under the distribution induced by a recurrent policy trained using recurrent Q-learning, its hidden state provide a high amount of information about the belief of relevant state variables, at any time step. The hidden state of the RNN is thus a statistic of the history that encodes information about the belief. In addition, at any time step, the network performs an update of this statistic, based on the actions and observations that are observed. The RNN thus implements a filter that provides a statistic encoding the belief.

However, it was only shown that the RNN produces such a statistic under the distribution of histories induced by the learned policy. For the sake of robustness of the policy to perturbations of histories, we might want this statistic to also provide information about the belief under other distribution of histories.

In Appendix 3.F, we propose an experimental protocol to study the generalization of the learned statistics. The results show that the MI between the hidden states and the beliefs also increases throughout the training process, under distributions induced by various ε-greedy policies, even the fully random policy. We impute those results to the following reasons. First, the DRQN algorithm approximates the Q-function, which generally requires a richer statistic of the history than the optimal policy. Second, the DRQN algorithm makes use of exploration, which allows the RNN to learn from histories that are diverse. However, we still observe that the higher the noise, the lower the MI. From these results, we conclude that the statistic that is learned by the network generalizes reasonably well to other distributions of histories.

3.5 Conclusion

In this work, we have shown empirically for several POMDPs that RNNs approximating the Q-function with a recurrent Q-learning algorithm [Hausknecht and Stone, 2015, Zhu et al., 2017] produces a statistic in their hidden states that provide a high amount of information about the belief of state variables that are relevant for optimal control. More precisely, we have shown that the MI between the hidden states of the RNN and the belief of states variables that are relevant for optimal control was increasing throughout the training process. In addition, we have shown that the ability of a recurrent architecture to reproduce, through a high MI, the belief filter conditions the performance of its policy. Finally, we showed that the MI between the hidden states and the beliefs of state variables that are irrelevant for optimal control decreases through the training process, suggesting that RNNs only focus on the relevant part of the belief.

This work also opens up several paths for future work. First, this work suggests that enforcing a high MI between the hidden states and the beliefs leads to an increase in the performances of the algorithm and in the return of the resulting policy. While other works have focused on an explicit representation of the belief in the hidden states [Karkus et al., 2017, Igl et al., 2018], which required to design specific recurrent architectures, we propose to implicitly embed the belief in the hidden state of any recurrent architecture by maximising their MI. When the belief or state particles are available, this can be done by adding an auxiliary loss such that the RNN also maximizes the MI. In practice, this can be implemented by backpropagating the MINE loss beyond the MINE architecture through the unrolled RNN architecture, such that the hidden states are optimized to get a higher MI with the beliefs.

Moreover, this work could be extended to algorithms that approximate other functions of the histories than the Q-function. Notably, this study could be extended to the hidden states of a recurrent policy learned by policy-gradient algorithms or to the hidden states of the actor and the critic in actor-critic methods. We may nevertheless expect to find similar results since the value function of a policy tends towards the optimal value function when the policy tends towards the optimal policy.

3.A Environments

In this section, the class of environments that are considered in this work are introduced. Then, the environments are formally defined.

3.A.1 Class of Environments

In the experiments, the class of POMDPs that are considered is restricted to those where we can observe from o_t if a state s_t is terminal. A state $s \in \mathcal{S}$ is said to be terminal if, and only if,

$$\begin{cases} T(s'|s,a) = \delta_s(s'),\ \forall s' \in \mathcal{S}, \forall a \in \mathcal{A}, & (3.24) \\ R(s,a,s') = 0,\ \forall a \in \mathcal{A}. & (3.25) \end{cases}$$

where δ_s denotes the Dirac distribution centred in $s \in \mathcal{S}$. As can be noted, the expected cumulative reward of any policy when starting in a terminal state is zero. As a consequence, the Q-function of a history for which we observe a terminal state is also zero for any initial action. The PRQL algorithm thus only has to learn the Q-function of histories that have not yet reached a terminal state. It implies that the histories that are generated in the POMDP can be interrupted as soon as a terminal state is observed.

3.A.2 T-Maze Environments

The T-Maze environment is a POMDP $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, P, \gamma)$ parameterized by the maze length $L \in \mathbb{N}$ and the stochasticity rate $\lambda \in [0, 1]$. The formal definition of this environment is given below.

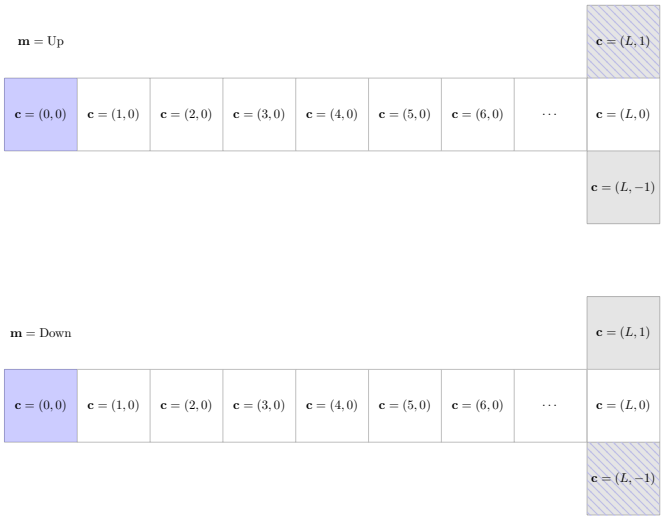


Figure 3.10: T-Maze state space. Initial states in blue, terminal states in grey, and treasure states hatched.

State space. The discrete state space $\mathcal{S}$ is composed of the set of positions $\mathcal{C}$ for the agent in each of the two maze layouts $\mathcal{M}$. The maze layout determines

the position of the treasure. Formally, we have,

$$\begin{cases} \mathcal{S} = \mathcal{M} \times \mathcal{C}, & (3.26) \\ \mathcal{M} = \{\text{Up}, \text{Down}\}, & (3.27) \\ \mathcal{C} = \{(0,0), \ldots, (L,0)\} \cup \{(L,1), (L,-1)\}. & (3.28) \end{cases}$$

A state $s_t \in \mathcal{S}$ is thus defined by $s_t = (m_t, c_t)$ with $m_t \in \mathcal{M}$ and $c_t \in \mathcal{C}$. Let us also define $\mathcal{F} = \{s_t = (m_t, c_t) \in \mathcal{S} | c_t \in \{(L,1), (L,-1)\}\}$ the set of terminal states, four in number.

Action space. The discrete action space $\mathcal{A}$ is composed of the four possible moves that the agent can take,

$$\mathcal{A} = \{(1,0), (0,1), (-1,0), (0,-1)\}, \tag{3.29}$$

that correspond to Right, Up, Left and Down, respectively.

Observation space. The discrete observation space $\mathcal{O}$ is composed of the four partial observations of the state that the agent can perceive,

$$\mathcal{O} = \{\text{Up}, \text{Down}, \text{Corridor}, \text{Junction}\}. \tag{3.30}$$

Initial state distribution. The two possible initial states are $s_0^{\text{Up}} = (\text{Up}, (0,0))$ and $s_0^{\text{Down}} = (\text{Down}, (0,0))$, depending on the maze in which the agent lies. The initial state distribution $P \colon \mathcal{S} \to [0,1]$ is thus given by,

$$P(s_0) = \begin{cases} 0.5 & \text{if } s_0 = s_0^{\text{Up}}, \\ 0.5 & \text{if } s_0 = s_0^{\text{Down}}, \\ 0 & \text{otherwise.} \end{cases} \tag{3.31}$$

Transition distribution. The transition distribution function $T \colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$ is given by,

$$T(s_{t+1}|s_t, a_t) = \begin{cases} \delta_{s_t}(s_{t+1}) & \text{if } s_t \in \mathcal{F}, \\ (1-\lambda)\delta_{f(s_t,a_t)}(s_{t+1}) + \frac{\lambda}{4}\left(\sum_{a \in \mathcal{A}} \delta_{f(s_t,a)}(s_{t+1})\right) & \text{otherwise,} \end{cases} \tag{3.32}$$

where $s_t \in \mathcal{S}, a_t \in \mathcal{A}$ and $s_{t+1} \in \mathcal{S}$, and f is given by,

$$f(s_t, a_t) = \begin{cases} s_{t+1} = (m_t, c_t + a_t) & \text{if } s_t \notin \mathcal{F}, c_t + a_t \in \mathcal{C}, \\ s_{t+1} = (m_t, c_t) & \text{otherwise,} \end{cases} \tag{3.33}$$

where $s_t = (m_t, c_t) \in \mathcal{S}$ and $a_t \in \mathcal{A}$.

Reward function. The reward function $R\colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is given by,

$$R(s_t, a_t, s_{t+1}) = \begin{cases} 0 & \text{if } s_t \in \mathcal{F}, \\ 0 & \text{if } s_t \notin \mathcal{F}, s_{t+1} \notin \mathcal{F}, s_t \neq s_{t+1}, \\ -0.1 & \text{if } s_t \notin \mathcal{F}, s_{t+1} \notin \mathcal{F}, s_t = s_{t+1}, \\ 4 & \text{if } s_t \notin \mathcal{F}, s_{t+1} \in \mathcal{F}, c_{t+1} = (L, 1), m_{t+1} = \text{Up}, \\ 4 & \text{if } s_t \notin \mathcal{F}, s_{t+1} \in \mathcal{F}, c_{t+1} = (L, -1), m_{t+1} = \text{Down}, \\ -0.1 & \text{if } s_t \notin \mathcal{F}, s_{t+1} \in \mathcal{F}, c_{t+1} = (L, -1), m_{t+1} = \text{Up}, \\ -0.1 & \text{if } s_t \notin \mathcal{F}, s_{t+1} \in \mathcal{F}, c_{t+1} = (L, +1), m_{t+1} = \text{Down}, \end{cases} \tag{3.34}$$

where $s_t = (m_t, c_t) \in \mathcal{S}, a_t \in \mathcal{A}$ and $s_{t+1} = (m_{t+1}, c_{t+1}) \in \mathcal{S}$.

Observation distribution. In the T-Maze, the observations are deterministic. The observation distribution $O\colon \mathcal{S} \times \mathcal{O} \to [0, 1]$ is given by,

$$O(o_t|s_t) = \begin{cases} 1 & \text{if } o_t = \text{Up}, c_t = (0, 0), m_t = \text{Up}, \\ 1 & \text{if } o_t = \text{Down}, c_t = (0, 0), m_t = \text{Down}, \\ 1 & \text{if } o_t = \text{Corridor}, c_t \in \{(1, 0), \ldots, (L-1, 0)\}, \\ 1 & \text{if } o_t = \text{Junction}, c_t \in \{(L, 0), (L, 1), (L, -1)\}, \\ 0 & \text{otherwise}, \end{cases} \tag{3.35}$$

where $s_t = (m_t, c_t) \in \mathcal{S}$ and $o_t \in \mathcal{O}$.

Exploration policy. The exploration policy $\mathcal{E}\colon \mathcal{A} \to [0, 1]$ is a stochastic policy that is given by $\mathcal{E}(\text{Right}) = 1/2$ and $\mathcal{E}(\text{Other}) = 1/6$ where Other $\in \{\text{Up}, \text{Left}, \text{Down}\}$. It enforces the exploration of the right hand side of the maze layouts. This exploration policy, tailored to the T-Maze environment, allows one to speed up the training procedure, without interfering with the study of this work.

Truncation horizon. The truncation horizon H of the DRQN algorithm is chosen such that the expected displacement of an agent moving according to the exploration policy in a T-Maze with an infinite corridor on both sides is greater than L. Let $r = \mathcal{E}(\text{Right})$ and $l = \mathcal{E}(\text{Left})$. In this infinite T-Maze, the probability of increasing its position is $p = (1 - \lambda)r + \lambda\frac{1}{4}$ and the probability of decreasing its position is $q = (1 - \lambda)l + \lambda\frac{1}{4}$. As a consequence, starting at 0, the expected displacement after one time step is $\bar{x}_1 = (1 - \lambda)(r - l)$. By independence, $\bar{x}_H = H\bar{x}_1$ such that, for $\bar{x}_H \geq L$, the time horizon is given by,

$$H = \left\lceil \frac{L}{(1 - \lambda)(r - l)} \right\rceil. \tag{3.36}$$

3.A.3 Mountain Hike Environments

The Varying Mountain Hike environment is a POMDP $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, P, \gamma)$ parameterized by the sensor variance $\sigma_O \in \mathbb{R}$ and the transition variance $\sigma_T \in \mathbb{R}$. The formal definition of this environment is given below.

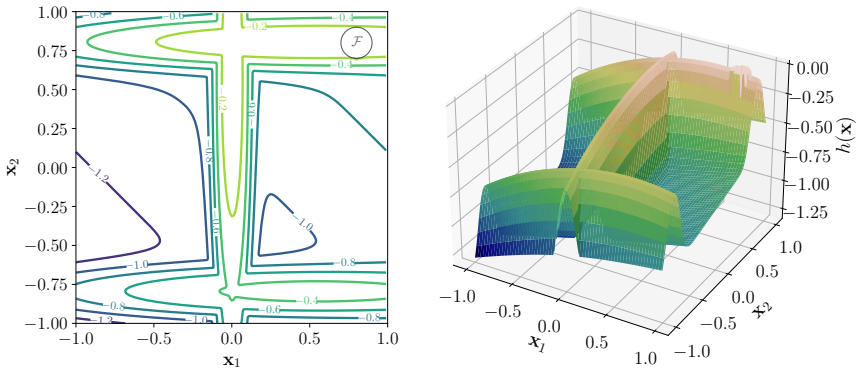


Figure 3.11: Mountain hike altitude function h in $\mathcal{X}$.

State space. The state space $\mathcal{S}$ is the set of positions $\mathcal{X}$ and orientations $\mathcal{C}$ that the agent can take. Formally, we have,

$$\begin{cases} \mathcal{S} = \mathcal{X} \times \mathcal{C}, & (3.37) \\ \mathcal{X} = [-1, 1]^2, & (3.38) \\ \mathcal{C} = \{0°, 90°, 180°, 270°\}. & (3.39) \end{cases}$$

The orientation $c = 0°, 90°, 180°$ and $270°$ corresponds to facing East, North, West and South, respectively. The set of terminal states is,

$$\mathcal{F} = \{s = (x, c) \in \mathcal{S} | \|x - (0.8, 0.8)\| < 0.1\}. \tag{3.40}$$

Action space. The discrete action space $\mathcal{A}$ is composed of the four possible directions in which the agent can move,

$$\mathcal{A} = \{(0, 0.1), (-0.1, 0), (0, -0.1), (0.1, 0)\}. \tag{3.41}$$

that correspond to Forward, Left, Backward and Right, respectively.

Observation space. The continuous observation space is $\mathcal{O} = \mathbb{R}$.

Initial state distribution. The initial position is always is always $x = (-0.8, -0.8)$ and the initial orientation is sampled uniformly in $\mathcal{C}$, such that the initial state distribution $P\colon \mathcal{S} \to [0, 1]$ is given by,

$$P(s_0) = \sum_{c \in \mathcal{C}} \frac{1}{|\mathcal{C}|} \delta_{((-0.8,-0.8),c)}(s_0). \tag{3.42}$$

Transition distribution. The transition distribution $T\colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is given by the conditional probability distribution of the random variable $(s_{t+1}|s_t, a_t)$ that is defined as,

$$s_{t+1} = \begin{cases} s_t & \text{if } s_t \in \mathcal{F}, \\ \text{clamp}_{\mathcal{S}}\left(s_t + R(c)\, a_t + \mathcal{N}(0, \sigma_T)\right) & \text{otherwise,} \end{cases} \tag{3.43}$$

where $\text{clamp}_{\mathcal{S}}(s)$ is the function that maps s to the point in $\mathcal{S}$ that minimizes its distance with s, and,

$$R(c) = \begin{pmatrix} \cos c & -\sin c \\ \sin c & \cos c \end{pmatrix}, \tag{3.44}$$

is the two-dimensional rotation matrix for an angle c.

Reward function. The reward function $R\colon \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is given by,

$$R(s_t, a_t, s_{t+1}) = \begin{cases} 0 & \text{if } s_t \in \mathcal{F}, \\ h(s_{t+1}) & \text{otherwise,} \end{cases} \tag{3.45}$$

where $s_t \in \mathcal{S}, a_t \in \mathcal{A}, s_{t+1} \in \mathcal{S}$, and $h\colon \mathcal{S} \to \mathbb{R}^-$ is the function that gives the relative altitude to the mountain top in any state. Note that the altitude is independent of the agent orientation.

Observation distribution. The observation distribution $O\colon \mathcal{S} \times \mathcal{O} \to [0, 1]$ is given by,

$$O(o_t|s_t) = \phi(o_t; h(s_t), \sigma_O^2), \tag{3.46}$$

where $s_t \in \mathcal{S}$ and $o_t \in \mathcal{O}$, and where $\phi(\cdot; \mu, \sigma^2)$ denotes the probability density function of a univariate Gaussian random variable with mean μ and standard deviation σ.

Mountain Hike. The Mountain Hike environment is a POMDP $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, P, \gamma)$, parameterized by the sensor variance $\sigma_O \in \mathbb{R}$ and the transition variance $\sigma_T \in \mathbb{R}$. The formal definition of this environment is identical to that of the Varying Mountain Hike, except that the initial orientation of the agent is always North, which makes it an easier problem. The initial state distribution is thus given by,

$$P(s_0) = \delta_{((-0.8,-0.8),90^\circ)}(s_0). \tag{3.47}$$

Exploration policy. The uniform distribution $\mathcal{U}(\mathcal{A})$ over the action space $\mathcal{A}$ is chosen as the exploration policy $\mathcal{E}(\mathcal{A})$.

Truncation horizon. The truncation horizon of the DRQN algorithm is chosen equal to $H = 80$ for the Mountain Hike environment and $H = 160$ for the Varying Mountain Hike environment.

3.B Deep Recurrent Q-learning

The DRQN algorithm is an instance of the PRQL algorithm that introduces several improvements over vanilla PRQL. First, it is adapted to the online setting by interleaving the generation of episodes and the update of the estimation Q_θ. In addition, in the DRQN algorithm, the episodes are generated with the ε-greedy policy $\sigma_\theta^\varepsilon\colon \mathcal{H} \to \Delta(\mathcal{A})$, derived from the current estimation Q_θ. This stochastic policy selects actions according to $\arg\max_{a \in \mathcal{A}} Q_\theta(\cdot, a)$ with probability $1 - \varepsilon$, and according to an exploration policy $\mathcal{E}(\mathcal{A}) \in \Delta(\mathcal{A})$ with probability

ε. In addition, a replay buffer of histories is used and the gradient is evaluated on a batch of histories sampled from this buffer. Furthermore, the parameters θ are updated with the Adam algorithm [Kingma and Ba, 2014]. Finally, the target $r_t + \gamma \max_{a \in \mathcal{A}} Q_{\theta'}(h_{t+1}, a)$ is computed using a past version $Q_{\theta'}$ of the estimation Q_θ with parameters θ' that are updated to θ less frequently, which eases the convergence towards the target, and ultimately towards the Q-function. The DRQN training procedure is detailed in Algorithm 3.1.

Algorithm 3.1: Deep recurrent Q-learning.

parameters: $N \in \mathbb{N}$ the buffer capacity,
$C \in \mathbb{N}$ the target update period (in episodes),
$E \in \mathbb{N}$ the number of episodes,
$H \in \mathbb{N}$ the truncation horizon,
$I \in \mathbb{N}$ the number of gradient steps after each episode,
$\varepsilon \in \mathbb{R}$ the exploration rate,
$\alpha \in \mathbb{R}$ the learning rate,
$B \in \mathbb{N}$ the batch size.

inputs: $\mathcal{E}(\mathcal{A}) \in \Delta(\mathcal{A})$ the exploration policy.

Initialize empty replay buffer $\mathcal{B}$.
Initialize parameters θ randomly.
for $e = 0, \dots, E-1$ **do**
  if $e \bmod C = 0$ **then**
    Update target network with $\theta' = \theta$.
  Draw an initial state s_0 according to P and observe o_0.
  Let $h_0 = (o_0)$.
  for $t = 0, \dots, H-1$ **do**
    Select $a_t \sim \mathcal{E}(\mathcal{A})$ with probability ε, otherwise select $a_t = \arg\max_{a \in \mathcal{A}} \{Q_\theta(h_t, a)\}$.
    Take action a_t and observe r_t and o_{t+1}.
    Let $h_{t+1} = (o_0, a_0, o_1, \dots, o_{t+1})$.
    if $|\mathcal{B}| < N$ **then** add $(h_t, a_t, r_t, o_{t+1}, h_{t+1})$ in replay buffer $\mathcal{B}$
    else replace oldest transition in replay buffer $\mathcal{B}$ by $(h_t, a_t, r_t, o_{t+1}, h_{t+1})$.
    if o_{t+1} *is terminal* **then**
      break
  for $i = 0, \dots, I-1$ **do**
    Sample B transitions $(h_t^b, a_t^b, r_t^b, o_{t+1}^b, h_{t+1}^b)$ uniformly from the replay buffer $\mathcal{B}$.
    Compute targets

$$y^b = \begin{cases} r_t^b + \gamma \max_{a \in \mathcal{A}} \left\{ Q_{\theta'}(h_{t+1}^b, a) \right\} & \text{if } o_{t+1}^b \text{ is not terminal,} \\ r_t^b & \text{otherwise.} \end{cases}$$

    Compute loss $L = \sum_{b=0}^{B-1} \left(y^b - Q_\theta(h_t^b, a_t^b) \right)^2$.
    Compute direction g using Adam optimizer.
    Perform gradient step $\theta = \theta + \alpha g$.

3.C Particle Filtering

As explained in Section 3.2, the belief filter becomes intractable for certain POMDPs. In particular, POMDPs with continuous state space require one to perform an integration over the state space. Furthermore, in these environments, the belief should be represented by a function over a continuous domain

instead of a finite-dimensional vector. Such arbitrary beliefs cannot be represented in a digital computer.

To overcome these two difficulties, the particle filtering algorithm proposes to represent an approximation of the belief by a finite set of samples that follows the belief distribution. In other words, we represent $b_t \in \Delta(\mathcal{S})$ by the set of M samples,

$$S_t = \{s_t^m\}_{m=0}^{M-1}, \tag{3.48}$$

where $s_t^m \in \mathcal{S}$, $m = 0, \dots, M-1$ are independent realizations of the belief distribution b_t.

Particle filtering is a procedure that allows one to sample a set of states S_t that follow the belief distribution b_t. The set is thus updated each time that a new action a_{t-1} is taken and a new observation o_t is observed. Although this procedure does not require to evaluate expression (3.8), it is necessary to be able to sample from the initial state distribution P and from the transition distribution T, and to be able to evaluate the observation distribution O. This process, illustrated in Algorithm 3.2, guarantees that the successive sets $S_0, \dots, S_H$ have (weighted) samples following the probability distribution $b_0, \dots, b_H$ defined by equation (3.8).

Algorithm 3.2: Particle filtering.

parameters: $M \in \mathbb{N}$ the number of particles
inputs: $H \in \mathbb{N}$ the number of transitions,
$h_H = (o_0, a_0, \dots, o_{H-1}, a_{H-1}, o_H) \in \mathcal{H}_H$ a history.

Sample $s_0^0, \dots, s_0^{M-1} \sim P$.
Let $h = 0$.
for $m = 0, \dots, M-1$ **do**
  Let $w_0^m = O(o_0|s_0^m)$.
  Let $h = h + w_0^m$.
for $m = 0, \dots, M-1$ **do**
  Let $w_0^m = w_0^m / h$.
Let $S_0 = \{(s_0^m, w_0^m)\}_{m=0}^{M-1}$.
for $t = 1, \dots, H$ **do**
  Let $h = 0$.
  for $m = 0, \dots, M-1$ **do**
    Sample $l \in \{0, \dots, M-1\}$ according to $p(l) = w_{t-1}^l$.
    Sample $s_t^m \sim T(\cdot|s_{t-1}^l, a_{t-1})$.
    Let $w_t^m = O(o_t|s_t^m)$.
    Let $h = h + w_t^m$.
  for $m = 0, \dots, M-1$ **do**
    Let $w_t^m = w_t^m / h$.
  Let $S_t = \{(s_t^m, w_t^m)\}_{m=0}^{M-1}$.
return *successive particles* $S_0, \dots, S_H$.

Algorithm 3.2 starts from N samples from the initial distribution P. These samples are initially weighted by their likelihood $O(o_0|s_0^n)$. Then, we have three steps that are repeated at each time step. First, the samples are resampled according to their weights. Then, given the action, the samples are updated by sampling from $T(\cdot|s_t^n, a_t)$. Finally, these new samples are weighted by their likelihood $O(o_{t+1}|s_{t+1}^n)$ given the new observation o_{t+1}, as for the initial samples.

As stated above, this method ensures that the (weighted) samples follow the distribution of the successive beliefs.

3.D Mutual Information Neural Estimator

In Subappendix 3.D.1, the MI estimator that is used in the experiments is formally defined, and the algorithm that is used to derive this estimator is detailed. In Subappendix 3.D.2, we formalize the extension of the MINE algorithm with the deep set architecture.

3.D.1 Mutual Information Estimation

As explained in Subsection 3.2.3, the ideal MI neural estimator, for a parameter space Φ, is given by,

$$I_\Phi(X;Y) = \sup_{\phi\in\Phi} i_\phi(X;Y), \tag{3.49}$$

$$i_\phi(X;Y) = \mathbb{E}_{z\sim p}\left[T_\phi(z)\right] - \log\left(\mathbb{E}_{z\sim q}\left[e^{T_\phi(z)}\right]\right). \tag{3.50}$$

However, both the estimation of the expectations and the computation of the supremum are intractable. In practice, the expectations are thus estimated with the empirical means over the set of samples $\{(x^n, y^n)\}_{n=0}^{N-1}$ drawn from the joint distribution p and the set of samples $\{(x^n, \tilde{y}^n)\}_{n=0}^{N-1}$ obtained by permuting the samples from Y, such that the pairs follow the product of marginal distributions $q = p_X \otimes p_Y$. In order to estimate the supremum over the parameter space Φ, the MINE algorithm proposes to maximize $i_\phi(X;Y)$ by stochastic gradient ascent over batches from the two sets of samples, as detailed in Algorithm 3.3. The final parameters ϕ^* obtained by this maximization procedure define the estimator,

$$\hat{I} = \frac{1}{N}\sum_{n=0}^{N-1} T_{\phi^*}(x^n, y^n) - \log\left(\frac{1}{N}\sum_{n=0}^{N-1} e^{T_{\phi^*}(x^n,\tilde{y}^n)}\right), \tag{3.51}$$

that is used in the experiments. This algorithm was initially proposed in [Belghazi et al., 2018].

3.D.2 Deep Sets

As explained in Subsection 3.4.1, the belief computation is intractable for environments with continuous state spaces. In the experiments, the belief of such environments is approximated by a set of particles $S = \{s^m\}_{m=1}^M$ that are guaranteed to follow the belief distribution, such that $s^m \sim b$, $\forall s^m \in S$ (see Appendix 3.C). Those particles could be used for constructing an approximation of the belief distribution, a problem known as density estimation. We nonetheless do not need an explicit estimate of this distribution. Instead, the particles can be directly consumed by the MINE network. In this case, the two sets of input samples of the MINE algorithm take the form,

$$\{(x^n, y^n)\}_{n=0}^{N-1} = \{(z^n, S^n)\}_{n=0}^{N-1} \tag{3.52}$$

$$= \left\{(z^n, \{s^{n,m}\}_{m=1}^M)\right\}_{n=0}^{N-1}. \tag{3.53}$$

Algorithm 3.3: Mutual information neural estimator optimization.

parameters: $E \in \mathbb{N}$ the number of episodes,
$B \in \mathbb{N}$ the batch size,
$\alpha \in \mathbb{R}$ the learning rate.

inputs: $N \in \mathbb{N}$ the number of samples,
$\mathcal{D} = \{(x^n, y^n)\}_{n=0}^{N-1}$ the set of samples from the joint distribution.

Initialize parameters ϕ randomly.
for $e = 0, \dots, E-1$ **do**
  Let p a random permutation of $\{0, \dots, N-1\}$.
  Let $\tilde{p}_1$ a random permutation of $\{0, \dots, N-1\}$.
  Let $\tilde{p}_2$ a random permutation of $\{0, \dots, N-1\}$.
  while $i = 0, \dots, \left\lfloor \frac{N}{B} \right\rfloor$ **do**
    Let $S = \left\{ (x^{p(k)}, y^{p(k)}) \right\}_{k=iB}^{(i+1)B-1}$ a batch of samples from the joint distribution.
    Let $\tilde{S} = \left\{ (x^{\tilde{p}_1(k)}, y^{\tilde{p}_2(k)}) \right\}_{k=iB}^{(i+1)B-1}$ a batch of samples from the product of marginal distributions.
    Evaluate the lower bound,
$$L(\phi) = \frac{1}{B} \sum_{(x,y) \in S} T_\phi(x, y) - \log \left(\frac{1}{B} \sum_{(\tilde{x}, \tilde{y}) \in \tilde{S}} e^{T_\phi(\tilde{x}, \tilde{y})} \right).$$
    Evaluate bias corrected gradients $G(\phi) = \tilde{\nabla}_\phi L(\phi)$.
    Update network parameters with $\phi = \phi + \alpha G(\phi)$.

In order to process particles from sets S^n as input of the neural network T_ϕ, we choose an architecture that guarantees its invariance to permutations of the particles. The deep set architecture [Zaheer et al., 2017], that is written as $\rho_\phi \left(\sum_{s \in S} \psi_\phi(s) \right)$, provides such guarantees. Moreover, this architecture is theoretically able to represent any function on sets, under the assumption of having representative enough mappings ρ_ϕ and ψ_ϕ and the additional assumption of using finite sets S when particles come from an uncountable set as in this work. The function T_ϕ is thus given by,

$$T_\phi(z, S) = \mu_\phi \left(z, \rho_\phi \left(\sum_{s \in S} \psi_\phi(s) \right) \right), \tag{3.54}$$

when the belief is approximated by a set of particles.

3.E Hyperparameters

The hyperparameters of the DRQN algorithm are given in Table 3.1 and the hyperparameters of the MINE algorithm are given in Table 3.2. The value of those hyperparameters have been chosen a priori, except for the number of episodes of the DRQN algorithm and the number of epochs of the MINE algorithm. These were chosen so as to ensure convergence of the policy return and the MINE lower bound, respectively. The parameters of the Mountain Hike and Varying Mountain Hike environments are given in Table 3.3.

Name	Value	Description
S	2	Number of RNN layers
D	1	Number of linear layers (no activation function)
H	32	Hidden state size
N	8192	Replay buffer capacity
C	10	Target update period in term of episodes
I	10	Number of gradient steps after each episode
ε	0.2	Exploration rate
B	32	Batch size
α	1×10^{-3}	Adam learning rate

Table 3.1: Deep recurrent Q-network architecture and training hyperparameters.

Name	Value	Description
L	2	Number of hidden layers
H	256	Hidden layer size
N	10 000	Training set size
E	200	Number of epochs
B	1024	Batch size
α	1×10^{-3}	Adam learning rate
R	16	Representation size for the deep set architecture
α	0.01	EMA rate for the bias corrected gradient

Table 3.2: Mutual information neural estimator architecture and training hyperparameters.

Name	Value	Description
σ_O	0.1	Standard deviation of the observation noise
σ_T	0.05	Standard deviation of the transition noise

Table 3.3: Mountain Hike and Varying Mountain Hike parameters.

3.F Generalization to Other Distributions of Histories

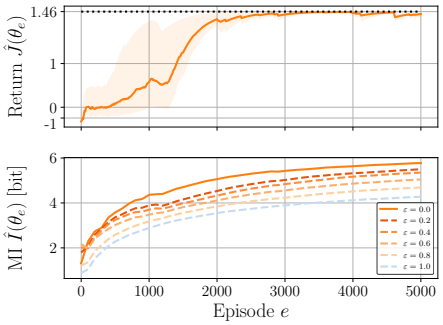
In this section, we study if the hidden state still provides information about the belief under other distributions of histories than the one induced by the learned policy (3.20). This generalization to other distributions is desirable for building policies that are more robust to perturbations of the histories.

We propose to study the evolution of the MI between the hidden state and the belief when adding noise to the policy used to sample the histories. Formally, instead of sampling the hidden states and beliefs according to (3.20), we propose to sample those according to,

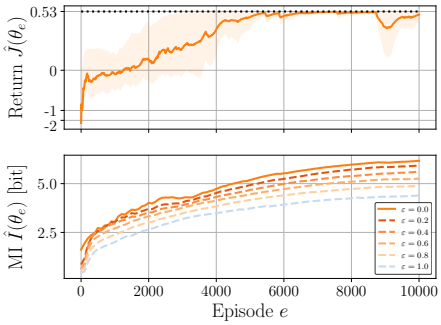
$$p_\varepsilon(z, b) = \sum_{t=0}^{\infty} p(t) \int_{\mathcal{H}} p(z, b|h)\, p_{\sigma_\theta^\varepsilon}(h|t)\, \mathrm{d}h, \tag{3.55}$$

where $p(t)$ is once again chosen to the uniform distribution over the time steps

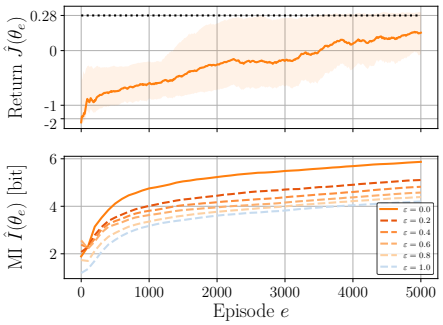
$p(t) = 1/H,\ t \in \{0, \dots, H-1\}$, $\sigma_\theta^\varepsilon$ is the ε-greedy policy as defined in Appendix 3.B, and $p_{\sigma_\theta^\varepsilon}(h|t)$ gives the conditional probability distribution induced by the policy $\sigma_\theta^\varepsilon$ over histories $h \in \mathcal{H}$ given that their length is $t \in \mathbb{N}_0$. Note that the training procedure remains unchanged.



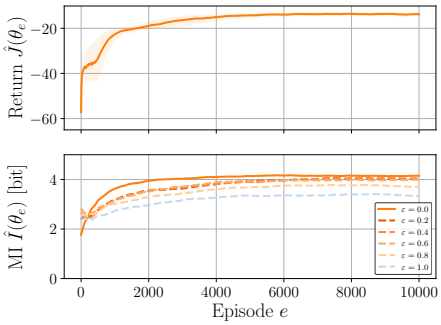
(a) Deterministic T-Maze ($L = 50$)



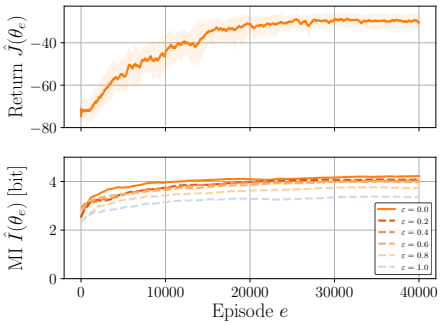
(b) Deterministic T-Maze ($L = 100$)



(c) Stochastic T-Maze ($L = 50$, $\lambda = 0.3$)



(d) Mountain Hike



(e) Varying Mountain Hike

Figure 3.12: Evolution of the return $\hat{J}(\theta_e)$ and the mutual information $\hat{I}(\theta_e)$ after e episodes, under distributions of histories induced by several ε-greedy policies, for the GRU cell. The maximal expected return is given by the dotted line.

The results of this additional study can be found in Figure 3.12, for $\varepsilon \in \{0.0, 0.2, 0.4, 0.6, 0.8, 1.0\}$. It can be noted that $p_{0.0}$ is the distribution of hidden states and beliefs induced by the learned policy (3.20), and $p_{1.0}$ is the distribution of hidden states and beliefs induced by a fully random policy. For reasons

of computational capacity, this analysis was carried out for the GRU cell only. This cell was chosen for being a standard cell that performs well in all environments in terms of return, unlike the LSTM. As can be seen in Figure 3.12, the MI between the hidden states and the beliefs increases throughout the training process, under all considered policies, even the fully random policy. We conclude that the correlation between the hidden states and beliefs generalizes reasonably well to other distributions. In other words, the hidden states still capture information about the beliefs even under other distributions of histories.

3.G Correlation Between Return and Mutual Information

The correlation between the empirical return and the estimated MI are computed with the Pearson's linear correlation coefficient and the Spearman's rank correlation coefficient. These coefficients are reported for all environments and all cells in Table 3.4 and Table 3.5. The columns named *all* give the correlation coefficients measured over all samples of $\hat{I}$ and $\hat{J}$ from all cells.

Environment	All	LSTM	GRU	BRC	nBRC	MGU
T-Maze ($L = 50$, $\lambda = 0.0$)	0.8233	0.7329	0.8500	0.8747	0.9314	0.9178
T-Maze ($L = 100$, $\lambda = 0.0$)	0.5347	0.3624	0.6162	0.6855	0.6504	0.6299
T-Maze ($L = 50$, $\lambda = 0.3$)	0.5460	0.2882	0.8008	0.7229	0.7424	0.6159
Mountain Hike	0.5948	0.7352	0.6177	0.4338	0.5857	0.5485
Varying Mountain Hike	0.5982	0.6712	0.4530	0.4446	0.3669	0.3006

Table 3.4: Pearson's linear correlation coefficients.

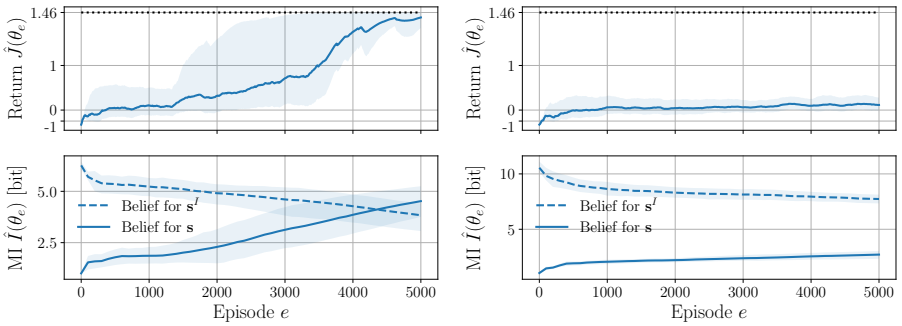
Environment	All	LSTM	GRU	BRC	nBRC	MGU
T-Maze ($L = 50$, $\lambda = 0.0$)	0.6419	0.7815	0.5963	0.5403	0.4009	0.5002
T-Maze ($L = 100$, $\lambda = 0.0$)	0.6666	0.5969	0.7108	0.5058	0.4605	0.5534
T-Maze ($L = 50$, $\lambda = 0.3$)	0.6403	0.3730	0.6600	0.5090	0.4706	0.6497
Mountain Hike	0.2965	0.5933	0.1443	0.2762	0.4337	0.2630
Varying Mountain Hike	0.6176	0.6869	0.3677	0.4355	0.2955	0.2266

Table 3.5: Spearman's rank correlation coefficients.

3.H Belief of Variables Irrelevant for Optimal Control

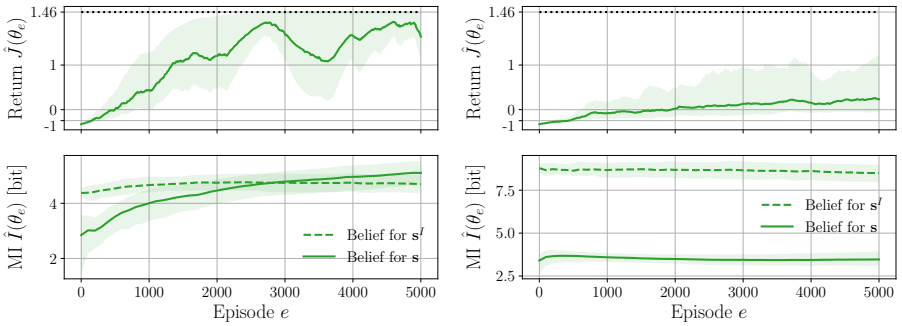
In this section, we report the evolution of the return and the MI between the hidden states and the belief of both the relevant and irrelevant variables for the LSTM, BRC, nBRC and MGU architectures. It completes the results obtained for the GRU cell in Subsection 3.4.4.

Figure 3.13, Figure 3.14, Figure 3.15, and Figure 3.16 show the evolution of the return and the MI for a T-Maze of length $L = 50$ with $d \in \{1, 4\}$ irrelevant state variables added to the process for these cells. These results are reported for the



(a) $d = 1$ (b) $d = 4$

Figure 3.13: Deterministic T-Maze ($L = 50$) with d irrelevant state variables. Evolution of the return $\hat{J}(\theta_e)$ and the mutual information $\hat{I}(\theta_e)$ for the belief of the irrelevant and relevant state variables after e episodes, for the LSTM cell.



(a) $d = 1$ (b) $d = 4$

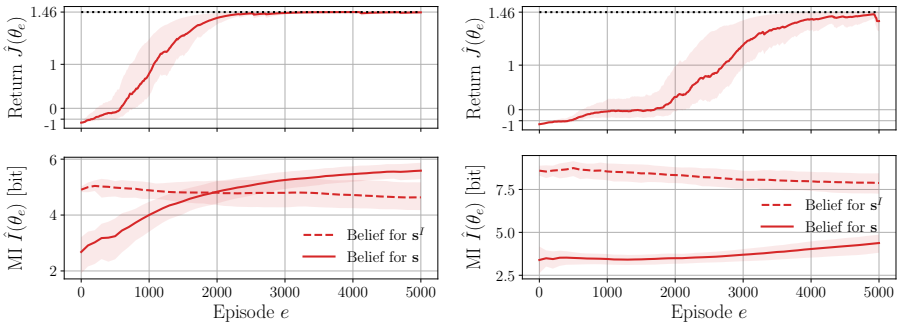
Figure 3.14: Deterministic T-Maze ($L = 50$) with d irrelevant state variables. Evolution of the return $\hat{J}(\theta_e)$ and the mutual information $\hat{I}(\theta_e)$ for the belief of the irrelevant and relevant state variables after e episodes, for the BRC cell.

GRU cell in Figure 3.8 (see Subsection 3.4.2). As can be seen from these figures, the return generally increases with the MI between the hidden states and the belief of state variables that are relevant for optimal control. Moreover, as for the GRU cell, the MI between the hidden states and the belief of irrelevant state variables generally decreases throughout the learning process.

Additionally, it can be observed that the LSTM and BRC cells fail in achieving a near-optimal return when $d = 4$. As far as the LSTM is concerned, it is reflected in its MI that reaches a lower value than the other RNNs. Likewise, the BRC cell does not reach a high return, and the MI does not increase at all. For this cell, it can be seen that the MI with the belief of irrelevant state variables is not decreasing, even with $d = 1$. The inability of the BRC cell to increase its MI with the belief of relevant variables and to decrease its MI with the belief of irrelevant variables might explain its bad performance in this environment.

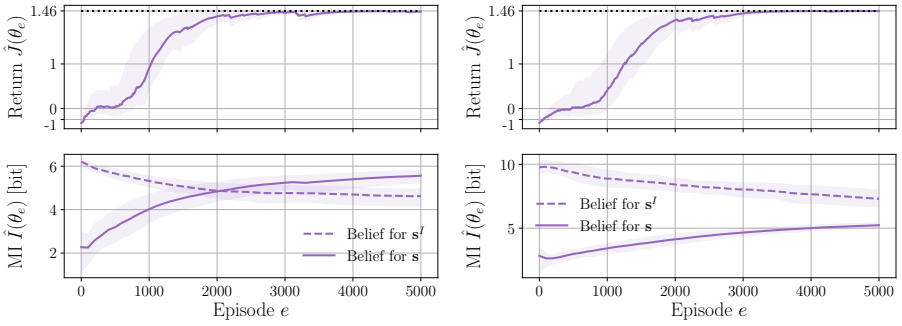
As far as the Mountain Hike is concerned, Figure 3.17, Figure 3.18, Figure 3.19 and Figure 3.20 show that all previous observations also hold for this environ-

ment with the LSTM, BRC, nBRC and MGU cells. These results are reported for the GRU cell in Figure 3.9 (see Subsection 3.4.2). As can be seen from these figures, the return clearly increases with the MI between the hidden states and the belief of relevant state variables, for all cells. In contrast, the MI with the belief of irrelevant state variables decreases throughout the learning process.



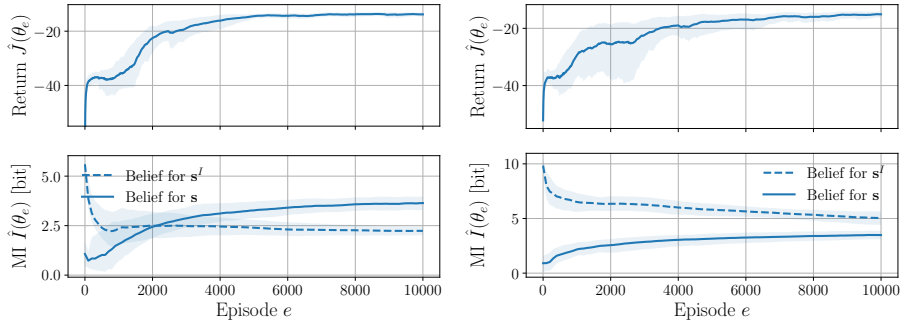
(a) $d = 1$ (b) $d = 4$

Figure 3.15: Deterministic T-Maze ($L = 50$) with d irrelevant state variables. Evolution of the return $\hat{J}(\theta_e)$ and the mutual information $\hat{I}(\theta_e)$ for the belief of the irrelevant and relevant state variables after e episodes, for the nBRC cell.



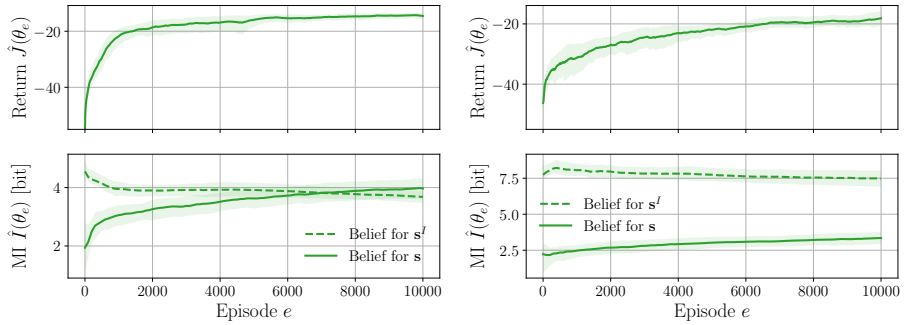
(a) $d = 1$ (b) $d = 4$

Figure 3.16: Deterministic T-Maze ($L = 50$) with d irrelevant state variables. Evolution of the return $\hat{J}(\theta_e)$ and the mutual information $\hat{I}(\theta_e)$ for the belief of the irrelevant and relevant state variables after e episodes, for the MGU cell.



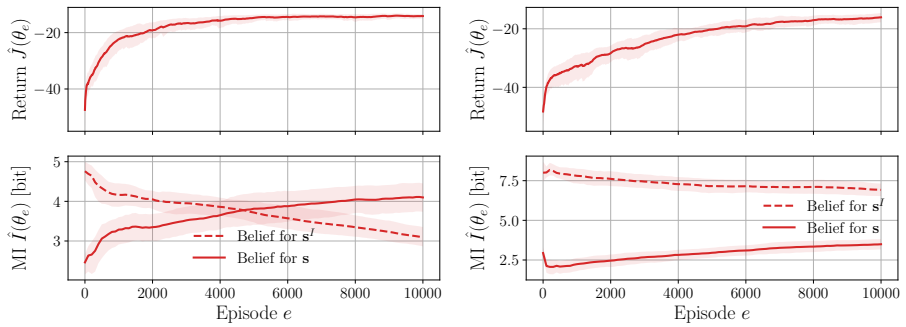
(a) $d = 1$ (b) $d = 4$

Figure 3.17: Mountain Hike with with d irrelevant state variables. Evolution of the return $\hat{J}(\theta_e)$ and the mutual information $\hat{I}(\theta_e)$ for the belief of the irrelevant and relevant state variables after e episodes, for the LSTM cell.



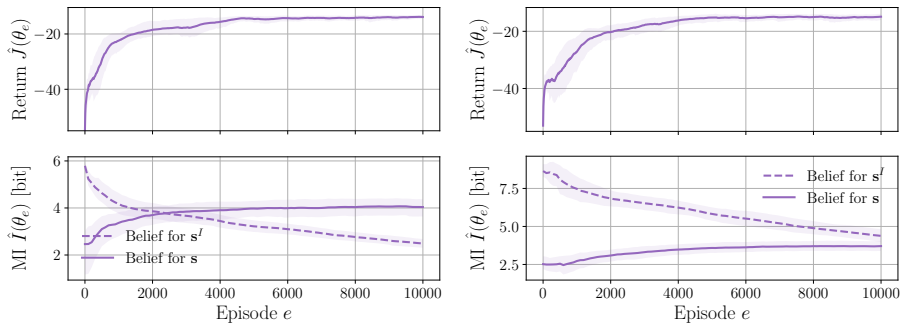
(a) $d = 1$ (b) $d = 4$

Figure 3.18: Mountain Hike with with d irrelevant state variables. Evolution of the return $\hat{J}(\theta_e)$ and the mutual information $\hat{I}(\theta_e)$ for the belief of the irrelevant and relevant state variables after e episodes, for the BRC cell.



(a) $d = 1$ (b) $d = 4$

Figure 3.19: Mountain Hike with with d irrelevant state variables. Evolution of the return $\hat{J}(\theta_e)$ and the mutual information $\hat{I}(\theta_e)$ for the belief of the irrelevant and relevant state variables after e episodes, for the nBRC cell.



(a) $d = 1$ (b) $d = 4$

Figure 3.20: Mountain Hike with with d irrelevant state variables. Evolution of the return $\hat{J}(\theta_e)$ and the mutual information $\hat{I}(\theta_e)$ for the belief of the irrelevant and relevant state variables after e episodes, for the MGU cell.

Chapter 4

Remembering for Learning

Warming Up Recurrent Neural Networks to Maximize Reachable Multistability Greatly Improves Learning. Gaspard Lambrechts, Florent De Geeter, Nicolas Vecoven, Guillaume Drion and Damien Ernst.

From the paper published in the *Neural Network* journal.

Abstract

Training recurrent neural networks is known to be difficult when time dependencies become long. In this work, we show that most standard cells only have one stable equilibrium at initialization, and that learning on tasks with long time dependencies generally occurs once the number of network stable equilibria increases, a property known as multistability. Multistability is often not easily attained by initially monostable networks, making learning of long time dependencies between inputs and outputs difficult. This insight leads to the design of a novel way to initialize any recurrent cell connectivity through a procedure called "warmup" to improve its capability to learn arbitrarily long time dependencies. This initialization procedure is designed to maximize network reachable multistability, which we define as the number of equilibria within the network that can be reached through relevant input trajectories, in few gradient steps. We show on several information restitution, sequence classification, and reinforcement learning benchmarks that warming up greatly improves learning speed and performance, for multiple recurrent cells, but sometimes impedes precision. We therefore introduce a double-layer architecture initialized with a partial warmup that is shown to greatly improve learning of long time dependencies while maintaining high levels of precision. This approach provides a general framework for improving learning abilities of any recurrent cell when long time dependencies are present. We also show empirically that other initialization and pretraining procedures from the literature implicitly foster reachable multistability of recurrent cells.

4.1 Introduction

Despite their performances and widespread use, recurrent neural networks (RNN) are known to be blackbox models with extremely complex internal dynamics. A growing body of work has focused on understanding the internal dynamics of trained RNNs [Sussillo and Barak, 2013, Ceni et al., 2020, Maheswaranathan et al., 2019], providing invaluable intuition into the RNN prediction process. This viewpoint has already been used to understand the difficulties for RNNs to capture longer time dependencies [Bengio et al., 1993, Doya, 1993]. In particular, recent work has highlighted the important role played by fixed points in RNN state spaces, that are defined as hidden states that updates to themself for a given input [Sussillo and Barak, 2013, Katz and Reggia, 2017]. This line of work has argued that locating such fixed points efficiently could provide insights into RNN dynamics and input-output properties. Here, we build upon this line of work by studying the impact of the number of reachable fixed points in an RNN on the ability to learn long time dependencies. Moreover, we highlight how maximizing the number of reachable fixed points at initialization can improve RNN learning, in particular in the presence of arbitrarily long dependencies.

More precisely, we introduce a fast-to-compute measure of the multistability of a network called variability amongst attractors (VAA). This measure gives the number of reachable attractors for a set of initial states. We show that loss decrease during learning in tasks with long time dependencies is highly correlated with an increase in VAA, highlighting both the relevance of the measure and the importance of multistability for efficient learning. Second, we use stochastic gradient ascent on a differentiable proxy of the VAA, called VAA*, as a way of maximizing the number of reachable attractors within the network at initialization. We show that this technique strongly improves performance on long time dependencies benchmarks, at the cost of precision, the latter relying on the richness of network transient dynamics. Third, we propose a parallel recurrent network structure with a partial warmup that enables one to combine long-term memory through multistability with precision through rich transient dynamics. Finally, we show empirically that other methods from the literature such as the chrono initialization and the bistable recurrent cells implicitly achieve the same goal of maximising the number of reachable attractors.

In Section 4.3, RNNs are introduced as dynamical systems and the concept of multistability is introduced for those systems. In Section 4.4, the supervised learning and reinforcement learning benchmarks are introduced. In Section 4.5, the VAA is introduced along with the estimation procedure of the multistability of an RNN for a set of initial states. The correlation between multistability and learning is shown empirically on the benchmarks with long time dependencies. In Section 4.6, the VAA* is introduced along with the warmup procedure that fosters multistability at initialization. The benefits of warmup are shown empirically on benchmarks with long time dependencies. In addition, the double-layer architecture is introduced and shown to achieve a better performance on all benchmarks. Finally, Section 4.7 concludes and proposes several future works.

4.2 Related Works

Training RNNs is known to be difficult when time dependencies become too long [Pascanu et al., 2013]. Indeed, the most used algorithm to train RNNs is the backpropagation through time (BPTT) algorithm [Werbos, 1990], which unrolls the RNN to see it as a feedforward neural network with shared weights before applying the backpropagation. However, the longer the sequence, the deeper the corresponding feedforward neural network is. Backpropagating through such deep networks often leads to vanishing or exploding gradients, and different methods have been proposed to tackle this issue. These methods usually act on one of three different levels: the training, the initialization/pretraining and the network architecture.

Training These methods modify the training of RNNs. For instance, clipping the gradients [Pascanu et al., 2013] prevents the gradients from exploding. Another example is the truncated variant of BPTT [Williams and Zipser, 2013], which does not propagate gradients through the whole sequences, but rather through parts of these sequences, leading to gradients that vanish or explode less often. It is likely that truncating the BPTT prevents from learning long time dependencies efficiently. Finally, Trinh et al. [2018] propose adding auxiliary losses at some time steps, to avoid having only one loss computed at the end of the sequences. These losses are computed in an unsupervised fashion: either a decoder has to reconstruct a part of the sequence (*reconstruction* loss), or a network has to predict the next input (*prediction* loss). This method can also be used as a pretraining to first train the RNN to encode correctly the sequences. This work achieved good results on very long sequences.

Initialization/pretraining The goal of these methods is to bring the network weights to a better place in the parameters space where the learning will be better and faster. Notably, the chrono-initialization [Tallec and Ollivier, 2018, Westhuizen and Lasenby, 2018] changes the initial biases parameters to improve the learning of long time dependencies. Some pretraining methods rely on autoencoders: Pasa and Sperduti [2014] use the parameters of a linear encoder as initial weights for the RNN, Sagheer and Kotb [2019] train a LSTM-based stacked autoencoder layer-wise before adding a output layer and fine-tuning on the dataset and Ong et al. [2014] introduce a dynamic pretraining of AE specifically made for time-series. Pasa et al. [2015] pretrain the RNN on a smoothed version of the dataset produced by a first-order hidden Markov model and then fine-tunes on the original dataset. Tang et al. [2016] first train a DNN before using it as a teacher to train the RNN. Ienco et al. [2019] focus on multi-class sequences classification. A trained RNN is used to rank the classes by decreasing order of complexity, then a new RNN is pretrained to predict the most complex class, then the second one, etc. All these pretraining methods have improved the performance of RNNs either on classification or on time-series prediction tasks. While making the final training of the network easier and better, none of them seems to directly promote the learning of long time dependencies.

Network architectures The most notable improvement made in the RNN architectures is the introduction of the gates, which are used to control the flow

of information in the network and to help the gradients to propagate through the time. These gates have led to the development of the long-short term memory (LSTM) [Hochreiter and Schmidhuber, 1997] and the gated recurrent unit (GRU) [Cho et al., 2014a], which are now the most used RNNs in practice. In the experiments, we also consider the minimal gated unit (MGU) [Zhou et al., 2016], a minimal design among gated recurrent units that only has one gate. Other approaches include the introduction of different time-scales inside the RNN. The segmented-memory RNN [Chen and Chaudhari, 2009] splits the sequences into segments and uses a two-layers RNN, where the first layer is reset at the end of each segment, while the second one is updated when a new segment begins. The hierarchical RNN [Hihi and Bengio, 1995], the hierarchical multiscale RNN [Chung et al., 2017] and the clockwork RNN (CW-RNN) [Koutnik et al., 2014] stack recurrent layers that are updated at different frequencies. The structurally constrained recurrent network (SCRN) [Mikolov et al., 2015] imposes some constraints on a subset of the recurrent weights, forcing some neurons hidden states to be slowly updated. The nonlinear autoregressive with exogenous inputs (NARX) RNN [Lin et al., 1996, Jr and Barreto, 2008] uses the n previous hidden states as inputs, making it a n^{th}-order RNN. Likewise, novel recurrent cell dynamics, such as the bistable recurrent cell (BRC) and the neuromodulated BRC (NBRC) [Vecoven et al., 2021], have been introduced to help tackle long time dependencies benchmarks. NBRCs were specifically designed to maximize reachability of cellular bistability, providing a way to create never-fading memory at the cellular level. These results highlighted how dynamics of untrained RNNs, i.e., at initialization, can strongly impact learning performance of RNNs. In this work, we extend this approach at the network level by maximizing multistability of any recurrent cell type prior to learning. To this end, we propose a novel RNN pretraining procedure called "warmup" that is designed to maximize the number of RNN attractors that can be reached from hidden states resulting from input sequences. Compared to pretraining methods, this method is very efficient since it only requires a few gradient steps before reaching a multistable regime for the RNN.

4.3 Background

In this section, RNNs are introduced as dynamical systems. The fixed points of these systems are defined, and the notions of attractors, reachable attractors and multistability are introduced.

4.3.1 Recurrent Neural Networks

RNNs are parametric function approximators that are often used to tackle problems with temporal structure. Indeed, RNNs process the inputs sequentially, exhibiting memory through hidden states that are outputted after each time step, and processed at the next time step along with the following input. These connections allow RNNs to memorize relevant information that should be captured over multiple time steps. More formally, an RNN architecture is defined by its update function f, its output function g and its initialization function h that are parameterized by a parameter vector $\theta \in \mathbb{R}^d$. Let $u_{1:T} = [u_1, \dots, u_T]$, with $T \in \mathbb{N}$ and $u_t \in \mathbb{R}^n$, an input sequence. RNNs maintain an internal

memory state x_t through an update rule $x_t = f(x_{t-1}, u_t; \theta)$ and output a value $o_t = g(x_t; \theta)$, where the initial hidden state $x_0 = h(\theta)$ is often chosen to be zero. We note that often, the output of the RNN is simply its hidden state x_t, i.e. g is the identity function. RNNs can be composed of only one recurrent layer, or they can be built with L layers that are linked sequentially through $u_t^i = o_t^{i-1}$ with $u_t^1 = u_t$ and $o_t = o_t^L$, where o_t^i denotes the output of layer i and u_t^i its input. In this case, each layer i has its own update function f^i, output function g^i and initialization function h^i. Backpropagation through time is used to train these networks where gradients are computed through the complete sequence via the hidden states [Werbos, 1990]. The following recurrent architectures are considered in the experiments: LSTM, GRU, BRC, NBRC, MGU. The specific update functions of those RNNs can be found in Appendix 4.A. In addition, we consider the chrono-initialized LSTM.

4.3.2 Fixed Points in Recurrent Neural Networks

Fixed points in u. In dynamical systems, fixed points are defined as points in the state space that map to themselves through the update function, for a given input u. For a system f, we say that a state x^* is a fixed point in u if and only if,

$$x^* = f(x^*, u). \tag{4.1}$$

Attractors in u. Fixed points can either be fully attractive (attractors), fully repulsive (repellors), or combine attractive and repulsive manifolds (saddle points). For a constant input u, the set of starting states for which the system converges to the fixed point x^* is called basin of attraction of x^* in u and is written as,

$$\mathcal{B}_{x^*}^u = \left\{ x \;\middle|\; \lim_{n\to\infty} f^n(x, u) = x^* \right\}, \tag{4.2}$$

$$\text{with } f^n(x,u) = \begin{cases} f\left(f^{n-1}(x,u), u\right) \text{ if } n > 1, \\ f(x,u) \text{ if } n = 1. \end{cases} \tag{4.3}$$

If the limit is not defined for some point x, then this point does not belong to any basin of attraction in u. Mathematically, x^* is an attractor in u if its basin of attraction in u, $\mathcal{B}_{x^*}^u$, has a positive measure.

Reachable attractors in u. In particular, we say that an attractor x^* in u is reachable from some state x if, and only if $x \in \mathcal{B}_{x^*}^u$.

Monostability and multistability in u. Given a set of states $\mathcal{X} = \{x_1, \ldots, x_n\}$, a system that has a unique reachable attractor in u for all states is said to be monostable in u for this set, whereas a system that has multiple reachable attractors in u is said to be multistable in u for this set. More formally, f is said to be monostable in u for $\mathcal{X}$ if, and only if, there exists a unique attractor x^*, such that $\forall x \in \mathcal{X},\ x \in \mathcal{B}_{x^*}^u$. On the contrary, f is said to be multistable in u is, and only if, there exists at least two attractors x_1^* and x_2^* such that $x_1^* \neq x_2^*$ and $\exists x_1, x_2 \in \mathcal{X},\ x_1 \in \mathcal{B}_{x_1^*}^u,\ x_2 \in \mathcal{B}_{x_2^*}^u$.

Recurrent neural networks. Due to their temporal nature and update rules, RNNs can be seen as discrete-time non-linear dynamical systems. Formally, given a parameter vector θ, the system f is given by the update function of the RNN, such that $f(x, u) = f(x, u; \theta)$. Since attractors correspond to network steady states, they are thought to be the allowing factor for RNNs to retain information over a long period of time [Pascanu et al., 2013, Sussillo and Barak, 2013, Maheswaranathan et al., 2019].

4.4 Benchmarks

In this section, the different benchmarks are introduced. First, the supervised learning tasks are introduced, including long-term information restitution benchmarks in Subsection 4.4.1 and sequence classification benchmarks in Subsection 4.4.2. In Subsection 4.4.3, a reinforcement learning benchmark with partially observable environment is introduced. This environment contains long time dependencies.

4.4.1 Long-Term Information Restitution Benchmarks

The benchmarks introduced in this subsection contain long time dependencies, and therefore require networks able to remember relevant information for a long period. For those benchmarks, RNNs are trained on a dataset of 40 000 sample sequences and evaluated on a dataset of 40 000 sample sequences. During training, 20% of the training set is used as a validation set.

Copy first input benchmark. In this benchmark, the network is presented with a one-dimensional sequence of T time steps $u_{1:T}$, where $u_t \sim \mathcal{N}(0, 1)$, $t = 1, \ldots, T$, and is tasked at approximating the target $y_T = u_1$. This benchmark thus consists of memorizing the initial input for T time steps. It allows one to measure the ability of recurrent architectures to bridge long time dependencies when the length T is large. Given the output o_T of the network, we seek to minimize the squared error $\mathcal{L}(o_T, y_T) = (o_T - y_T)^2$.

Denoising benchmark. In this benchmark, the network is presented with a two-dimensional sequence of T time steps. The first dimension is a noised input stream $u^1_{1:T}$, where $u^1_t \sim \mathcal{N}(0, 1)$, $t = 1, \ldots, T$. Five time steps of this stream should be remembered and outputted one by one by the network at time steps $\{T - 4, \ldots, T\}$. These five time steps $\mathcal{S} = \{t_1, t_2, t_3, t_4, t_5\}$, with $t_1 < t_2 < t_3 < t_4 < t_5$, are sampled without replacement in $\{1, \ldots, T - N\}$ with $N \geq 5$. N is a hyperparameter that allows one to tune how long the network should be able to retain the information at a minimum. The five time steps are communicated to the network through the second dimension of the input $u^2_{1:T}$, where $u^2_t = 1$ if $t \in \mathcal{S}$, and $u^2_t = 0$ otherwise, for $t = 1, \ldots, T$. The target is thus given by $y_{T-4:T} = [u_{t_1}, u_{t_2}, u_{t_3}, u_{t_4}, u_{t_5}]$. Given the output sequence $o_{T-4:T}$ of the network, we seek to minimize the mean squared error $\mathcal{L}(o_{T-4:T}, y_{T-4:T}) = \sum_{t=T-4}^{T} (o_t - y_t)^2$.

4.4.2 Sequence Classification Benchmarks

The benchmarks introduced in this subsection are sequence classification problems and therefore require networks able to use the information received in the sequence in order to infer the class. For those benchmarks, RNNs are trained on datasets derived from the usual train and test sets of the original MNIST dataset. During training, 20% of the training set is used as a validation set.

Permuted sequential MNIST. In this benchmark, the network is presented with the MNIST images, where pixels are presented to the network one by one as a sequence of length $T = 28 \times 28 = 784$. It differs from the regular sequential MNIST in that pixels are shuffled in a random order. Note that all images are shuffled according to the same random order.[1] The network is tasked at outputting a probability for each possible digit that could be represented in the initial image. This benchmark is known to be a more complex challenge than the regular one. Given the output $o_T \in \mathbb{R}^{10}$ of the network and the true digit index $y_T \in \{1, \dots, 10\}$, we seek to minimize the negative log likelihood loss $\mathcal{L}(o_T, y_T) = -\log(o_T^{y_T})$.

Permuted line-sequential MNIST. This benchmark is the same as the permuted sequential MNIST benchmark, except that the pixels are fed 28 by 28, which corresponds to one line of the permuted image.[1] The input dimension is thus 28 instead of one. N black lines are added at the end of the sequence such that the total length of the sequence is $T = 28 + N$. This has the effect of a forgetting period, such that any relevant information for predicting the class probabilities will be farther from the prediction time step T.

4.4.3 Reinforcement Learning Benchmark

In reinforcement learning, the function approximators process sequences as input when considering partially observable Markov decision processes (POMDP). Indeed, in such environments, the optimal policies, as well as the value functions, are functions of the complete sequence of observations and past actions, called the history. In this work, we focus on the approximation of the history-action value function, or Q-function, in order to derive a near-optimal policy in the considered POMDP. The deep recurrent Q-network (DRQN) algorithm is used to approximate this Q-function with an RNN. From this approximation, we derive the fully greedy policy by taking the action that maximizes the Q-function for any given history. See Appendix 4.B for the definition of POMDPs and their Q-functions, and see Appendix 4.C for the detailed DRQN algorithm.

The partially observable environment that is considered is the T-Maze environment [Bakker, 2001]. The T-Maze is a POMDP where the agent is tasked with finding the treasure in a T-shaped maze (see Figure 4.1). The state is given by the position of the agent in the maze and the maze layout that indicates whether the goal lies up or down after the crossroads. The initial state determines the maze layout, and it never changes afterwards. The initial observation made by the agent indicates the layout. Navigating in the maze provides zero reward,

[1] The permutation is given by the following command in NumPy 1.23.2: `np.random.seed(42); np.random.permutation(28*28)`.

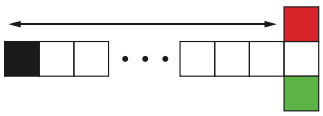


Figure 4.1: T-Maze layout example, with the initial position of the agent in black, the treasure in green and the cell to avoid in red.

except when bouncing onto a wall, in which case a reward of -0.1 is received. While traveling along the maze, the agent only receives the information that it has not yet reached the junction. Once the junction reached, the agent is notified: it must now choose a direction depending on the past information it remembers. Finding the treasure provides a reward of 4. Passed the crossroads, the states are always terminal. The optimal policy thus consists of going through the maze, while remembering the initial observation in order to take the correct direction at the crossroads. This POMDP is parameterized by the corridor length $L \in \mathbb{N}$ that determines the number of time steps for which the agent should remember the initial observation. The discount factor is $\gamma = 0.98$. This POMDP is formally defined in [Lambrechts et al., 2022].

4.5 Correlating Multistability and Learning

This section aims at showing the correlation that exists between multistability properties of RNNs and their ability to learn long time dependencies. To this end, in Subsection 4.5.1 we first introduce the VAA, a measure of the number of basins of attraction that are spanned by a set of states. In Subsection 4.5.2, we show how to estimate the multistability of an RNN using VAA by estimating the number of reachable attractors for a set of states resulting from the input sequences. We then carry out a number of experiments in Subsection 4.5.3 to show the correlation between multistability and learning with different types of RNN on the benchmarks previously introduced.

4.5.1 Variability Amongst Attractors

One way to quantify the multistability in u of a system for a set of states $\mathcal{X}$ is to count the number of different attractors that can be reached starting from those states. We name this measure variability amongst attractors (VAA). Formally, the VAA of a system f for a set of initial states $\mathcal{X}$ and an input u is defined as,

$$\mathrm{VAA}(f, \mathcal{X}, u) = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} \frac{1}{\sum_{j=1}^{|\mathcal{X}|} \delta\left(\limsup_{n\to\infty} \|f^n(x_i, u) - f^n(x_j, u)\| = 0\right)}, \tag{4.4}$$

where $\delta(x)$ is the Kronecker delta function that returns 1 when condition x is met, and 0 otherwise. It can be noted that this definition does not exclude limit cycles and considers states that are on the same limit cycle but far from each others as different attractors. This is a limitation that we discuss in our conclusion. In the following, we make the hypothesis that such limit cycles are not encountered in practice.

The denominator of Equation 4.4 gives the number of states in $\mathcal{X}$ that converge towards the same attractor as x_i. The sum of this fraction over all states that converge towards a given attractor is thus equal to one, such that the sum of this fraction over all states gives the number of different attractors. $\mathrm{VAA}(f, \mathcal{X}, u)$ is thus equal to the number of different attractors in u reached from the initial states contained in $\mathcal{X}$ divided by the number of initial states $|\mathcal{X}|$. Its maximal value is thus 1, when all reached attractors are different, and its minimal value is $\frac{1}{|\mathcal{X}|}$, when all the states have converged towards the same attractor (i.e., the system is monostable).

In practice, since it is impossible to evaluate the limits to infinity in the VAA, we fix a finite number of time steps M for state convergence, called the stabilization period. As a consequence, the system may not have completely converged towards the attractor after this period. We thus define a tolerance ε below which two final states are considered to correspond to the same attractor. This truncated VAA is written as,

$$\mathrm{VAA}_{M,\varepsilon}(f, \mathcal{X}, u) = \frac{1}{|\mathcal{X}|} \sum_{i=1}^{|\mathcal{X}|} \frac{1}{\sum_{j=1}^{|\mathcal{X}|} \delta\left(\|f^M(x_i, u) - f^M(x_j, u)\| \leq \varepsilon\right)}. \tag{4.5}$$

4.5.2 Estimating the Multistability of a Recurrent Neural Network

RNNs can exhibit a long-lasting memory through multistability in their hidden states [Vecoven et al., 2021]. Indeed, having multiple attractors that are reachable from different input sequences probably allows one to encode information about these sequences over the long term. We propose estimating the multistability of an RNN for a set of input sequences by computing the number of different reachable attractors for hidden states resulting from different input sequences. More precisely, we propose to compute $\mathrm{VAA}(f, \mathcal{X}, u)$ for hidden states $\mathcal{X}$ sampled from different input sequences. In practice, it is not feasible to estimate the VAA for all hidden states resulting from the set of input sequences. Indeed, computing the VAA is quadratic in the number of hidden states because of the pairwise distances. We thus propose to estimate the VAA by averaging its value over several small batches of hidden states sampled at random time steps in different sequences sampled from the set of input sequences. Moreover, we still have to choose the stable input u according to which we want to measure the multistability in u. In order to measure the multistability of the network for a wide range of stable inputs, we propose to measure the multistability on average for several inputs sampled according to a standard normal distribution. Note that for each batch of hidden states, a unique $u \sim \mathcal{N}(0, 1)$ is sampled and kept constant during the convergence period of M time steps. The resulting procedure for estimating the multistability of an RNN for a set of input sequences is given in Algorithm 4.1.

4.5.3 Experiments

In this subsection, we observe how the multistability of RNNs evolves when they are trained on the long-term information restitution and reinforcement learning

Algorithm 4.1: Multistability estimation.

parameters: $I \in \mathbb{N}$ the number of iterations to compute the average VAA,
$M \in \mathbb{N}$ the stabilization period,
$\varepsilon \in \mathbb{R}^+$ tolerance when considering state similarity,
inputs: $\mathcal{D} = \{u^1_{1:T_1}, \dots, u^N_{1:T_n}\}$ a set of N input sequences,
$\theta \in \mathbb{R}^{d_\theta}$ the parameters of the network.

Let $f = f(\cdot, \cdot;\ \theta)$ the dynamical system.
Initialize mean value $\overline{\mathrm{VAA}} = 0$.
for $i = 1, \dots, I$ **do**
  Sample a batch of input sequences $\mathcal{B} \sim \mathcal{D}$.
  Sample a random hidden state in each input sequence $\mathcal{X} = \mathrm{RandomHiddenStates}(\mathcal{B}, \theta)$.
  Sample $u \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$.
  $\overline{\mathrm{VAA}} = \overline{\mathrm{VAA}} + \frac{1}{I}\,\mathrm{VAA}_{M,\varepsilon}(f, \mathcal{X}, u)$.
return *average VAA* $\overline{VAA}$.

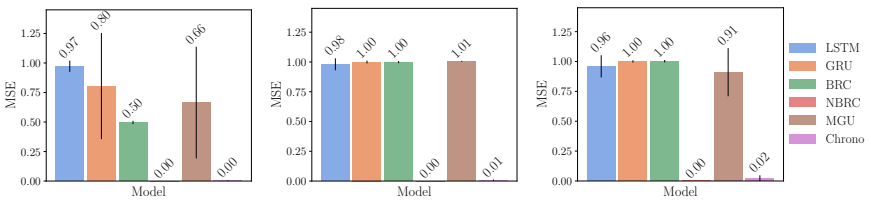
Algorithm 4.2: Random hidden states sampling.

inputs: $\mathcal{B} = \{u^1_{1:T_1}, \dots, u^n_{1:T_n}\}$ a batch of n input sequence sampled in the training set,
$\theta \in \mathbb{R}^{d_\theta}$ the parameters of the network.

Let $\mathcal{X} = \{\}$.
foreach $u^i_{1:T_i} \in \mathcal{B}$ **do**
  Sample a time step $t \sim \mathcal{U}(\{1, \dots, T_i\})$.
  Set $x^i_0 = h(\theta)$ where h is the initialization function of the RNN.
  for $k = 1, \dots, t$ **do**
    Set $x^i_k = f(x^i_{k-1}, u^i_k;\ \theta)$ where f is the update function of the RNN.
  Update $\mathcal{X} = \mathcal{X} \bigcup \left\{x^i_t\right\}$.
return *set of* n *hidden states* $\mathcal{X}$.

benchmarks introduced in Section 4.4. The multistability of these networks is estimated throughout the training procedure, using Algorithm 4.1. For the copy first input benchmark, networks are made up of one 128 neurons recurrent layer. For the other benchmarks, networks are made up of two recurrent layers, each of 256 neurons. All averages and standard deviations reported were computed over five different training sessions. Training was done using the Adam optimizer [Kingma and Ba, 2014] with a learning rate of 1×10^{-3} and a batch size of 32. All hyperparameters have been chosen a priori to standard values and are kept fixed. The goal here is not to measure the best performance of each architecture but rather to study, for a given architecture and optimization procedure, whether there is a link between learning and multistability for different benchmarks. In Subappendix 4.D.1, we show that those results also hold with other hyperparameters for the copy first input benchmark. In all experiments, the multistability is estimated with $M = 10\,000$, $\varepsilon = 1 \times 10^{-4}$, and $I = 10$.

Copy first input benchmark Figure 4.2 shows the performance of the different cells on this benchmark for different sequence lengths $T \in \{50, 300, 600\}$. The best-performing cell is the NBRC, whose performance is not affected by the length of the sequences. In comparison, the classical cells, MGU, LSTM and GRU, struggle to decrease their losses. Surprisingly, the BRC, a bistable cell, does not succeed in decreasing its loss. Generally speaking, the longer the sequences are, the worse their performances are. The last cell, the chrono-initialized LSTM, competes with the NBRC with its hyperparameter T_{max} chosen to 600. Figure 4.3 illustrates the correlation between the VAA and the validation loss for the LSTM and CHRONO cells. The LSTM cell, whose VAA increases late and little, fails to learn. On the other hand, the chrono initialized LSTM cell sees its loss decreasing while its VAA increases. This figure also shows that the chrono initialization promotes the learning of long time dependencies through multistability. Figure 4.4 illustrates the correlation between the VAA and the validation loss for the other cells. It is clear from this figure that the bistability mechanism introduced in the BRC and NBRC cells also promote multistability. Moreover, as for the LSTMs and chrono-initialized LSTMs, the loss only starts decreasing when the VAA increases.



(a) $T = 50$ (b) $T = 300$ (c) $T = 600$

Figure 4.2: Test MSE loss for the copy first input benchmark with different sequence lengths T. Mean and standard deviation are reported after 50 epochs.

Denoising benchmark Figure 4.5 shows the performance of the different cells on this benchmark for different forgetting periods $N \in \{5, 100\}$. Once again, the NBRC has the best performance, closely followed by the chrono-initialized LSTM. On this benchmark, the BRC also reaches a very low loss.

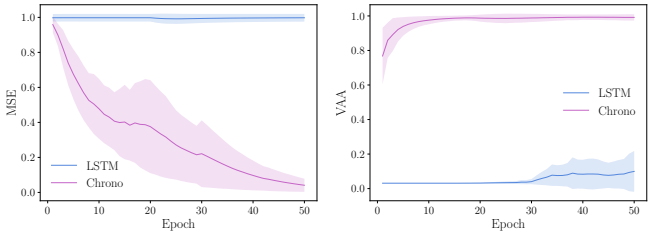


Figure 4.3: Evolution of the validation loss (left) and of the VAA (right) of LSTM networks, with and without chrono initialization, for the copy first input benchmark with $T = 50$. Mean and standard deviation are reported after 50 epochs.

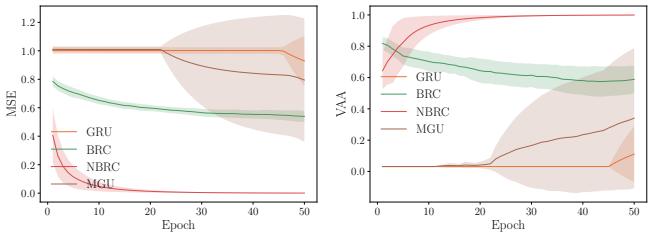
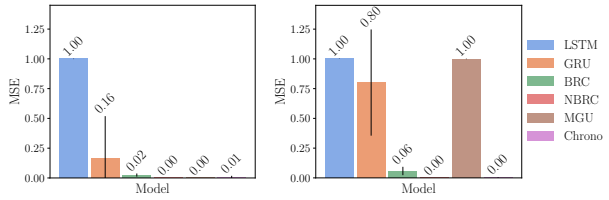


Figure 4.4: Evolution of the validation loss (left) and of the VAA (right) of GRU, MGU, BRC and NBRC networks, for the copy first input benchmark with $T = 50$. Mean and standard deviation are reported after 50 epochs.

Once again, we can see that all classical cells (LSTM, GRU, and MGU) generally fail in learning when longer time dependencies are present ($N = 100$). Figure 4.6 shows the evolution of the VAA and the validation loss of multiple LSTM cells, with and without chrono initialization, during the training on this benchmark. As for the previous benchmark, only the chrono-initialized LSTMs have a high VAA and efficiently decrease their loss. It can be noted that classically initialized LSTMs have a VAA close to zero throughout the learning on this harder benchmark. Figure 4.7 shows these results for the GRU, BRC, NBRC and MGU cells. It is observed that the GRU network has a very low VAA, and learning does not start before its VAA increases. The MGU network does not manage to learn on this benchmark while its VAA only slowly increases at the end of the training procedure. As far as the bistable networks (BRC and NBRC) are concerned, their VAA is directly maximized and learning starts directly, indicating that those indeed promote the learning of long time dependencies through multistability. Finally, Figure 4.8 shows the validation loss and the VAA of five different trainings of the GRU cell on the denoising benchmark with $N = 5$. It is clear that the GRU cell only starts decreasing its loss when its VAA has started increasing. This proves once more the correlation between the VAA and the learning on long-term information restitution benchmarks.

T-Maze benchmark In this reinforcement learning setting, a policy is derived from the approximation of the Q-function. The hyperparameters of the DRQN algorithm used for approximating the Q-function are given in Appendix 4.C.



(a) $N = 5$ (b) $N = 100$

Figure 4.5: Test MSE loss for the denoising benchmark with different forgetting periods N and $T = 200$. Mean and standard deviation are reported after 50 epochs.

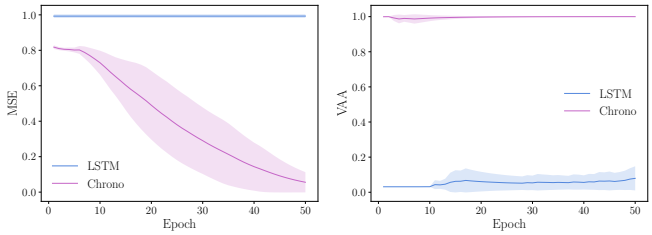


Figure 4.6: Evolution of the validation loss (left) and of the VAA (right) of LSTM networks, with and without chrono initialization, for the denoising benchmark with $N = 100$ and $T = 200$. Mean and standard deviation are reported after 50 epochs.

On the left in Figure 4.9, we can see the mean non-discounted cumulative reward obtained by the policies derived from GRU cells approximating the Q-function. On the right in Figure 4.9, we can see the VAA of these cells estimated with Algorithm 4.1 using the histories of the replay buffer as input sequences. Those value are clearly correlated. Indeed, the better the agent plays, the higher its VAA is.

4.6 Fostering Multistability at Initialization

In Subsection 4.6.1, we describe the warmup initialization procedure that allows one to maximize the estimated multistability of a network for a dataset of input sequences. Then, in Subsection 4.6.2, we compare classic cells to warmed-up cells on information restitution, sequence classification, and RL benchmarks and show the benefits of the warmup in tasks with long time dependencies, when considering the same standard hyperparameters as in the previous section. However, we also show that the warmup procedure does not improve the results in the sequence classification tasks. In Subsection 4.6.3, we introduce the double-layer architecture, that has both multistable and transient dynamics. We show that this architecture reaches a better performance both on information restitution and sequence classification benchmarks. Finally, in Subsection 4.6.4, we show that the advantage of the warmup and the double-layer architecture, shown for standard hyperparameters in Subsection 4.6.2 and Subsection 4.6.3,

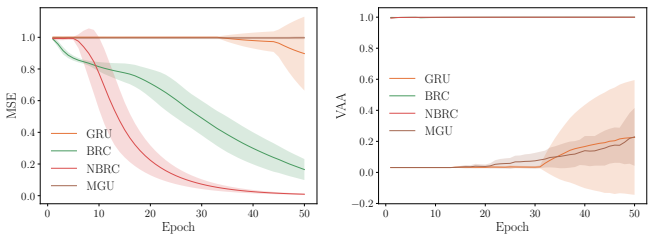


Figure 4.7: Evolution of the validation loss (left) and of the VAA (right) of GRU, MGU, BRC and NBRC networks, for the denoising benchmark with $N = 100$ and $T = 200$. Mean and standard deviation are reported after 50 epochs.

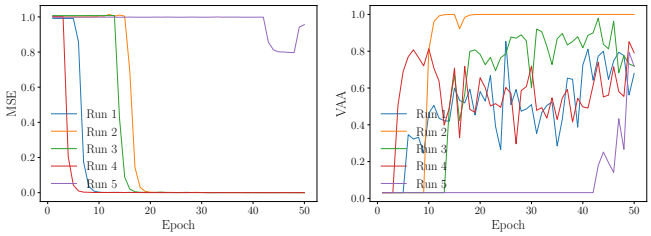


Figure 4.8: Evolution of the validation loss (left) and of the VAA (right) of multiple GRU networks, for the denoising benchmark with $N = 5$ and $T = 200$. Mean and standard deviation are reported after 50 epochs. Loss decrease only start when the network becomes multistable (VAA greater than $\frac{1}{|\mathcal{X}|}$).

also holds when optimizing the hyperparameters for each cell version (number of recurrent layers L, number of hidden units H, batch size B, learning rate α).

4.6.1 Warming Up Recurrent Neural Networks

The previous observations, that show a correlation between the multistability of a network and its ability to learn long time dependencies, suggest that fostering multistability could ease learning in this case. In order to promote the multistability of a network, we propose maximizing the number of reachable attractors for hidden states resulting from the set of input sequences. As for the estimation of the multistability, computing the VAA for all hidden states is not feasible because of its quadratic complexity. In practice, we propose using stochastic gradient descend (SGD) to maximize the number of reachable attractors for batches of hidden states from different input sequences. As for the estimation of the multistability, we sample a different stable input $u \sim \mathcal{N}(0, 1)$ for each batch of hidden states. We note however that SGD cannot be used directly on the estimation of the proportion of reachable attractors detailed in Algorithm 4.1, for two different reasons. First, the VAA and the $\text{VAA}_{M,\varepsilon}$ are not differentiable because of the Kronecker delta, which prevents from computing the gradient. Second, it is likely that hidden states convergence is slow when several RNNs are stacked. Indeed, the first layers must have reached stability for the following one to receive a stable input.

In order to solve the first problem, we introduce a differentiable proxy $\text{VAA}^*_{M,\varepsilon}$

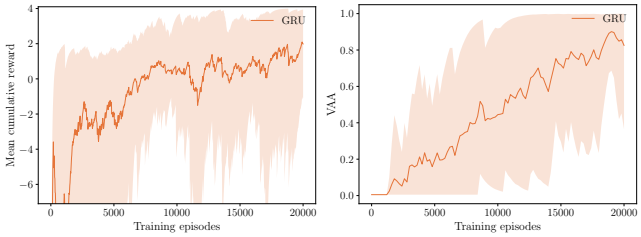


Figure 4.9: Evolution of the mean cumulative reward (left) and their VAA (right) obtained by GRU agents during DRQN training on a T-Maze of length 200. Mean and standard deviation are estimated over 3 training sessions.

of the $\text{VAA}_{M,\varepsilon}$. Instead of the denominator?

$$C_{i,j} = \delta\left(\left\|f^M(x_i,u) - f^M(x_j,u)\right\| \leq \varepsilon\right), \tag{4.6}$$

that is equal to 1 when the final states after truncated convergence are close enough, we use,

$$C^*_{i,j} = 1 - \frac{\max(0, \left\|\tanh f^M(x_i,u) - \tanh f^M(x_j,u)\right\| - \varepsilon)}{\|\tanh f^M(x_i,u) - \tanh f^M(x_j,u)\|}. \tag{4.7}$$

We note that the value of $C^*_{i,j}$ is strictly equal to 1 if $f^M(x_i,u)$ is close enough in Euclidian distance to $f^M(x_j,u)$. On the other hand, $C^*_{i,j}$ will be close to 0 when they are far away. We also note that $C^*_{i,j}$ will never be strictly equal to 0, but will get closer as the distance increases, since the fraction tends towards 1. It can be noted that we are not interested in states being far apart from each other, but just in them being different. However, we noticed in the experiments that this small bias provides a good direction for the gradient in order to reach multistability. For this reason, we need to apply a saturating function (hyperbolic tangent in this case) to the states in order to avoid extreme states when maximizing VAA^*. The resulting differentiable proxy of the VAA is given by,

$$\text{VAA}^*_{M,\varepsilon}(f,\mathcal{X},u) = \frac{1}{|\mathcal{X}|}\sum_{i=1}^{|\mathcal{X}|} \frac{1}{\sum_{j=1}^{|\mathcal{X}|} 1 - \frac{\max(0,\|\tanh f^M(x_i,u) - \tanh f^M(x_j,u)\| - \varepsilon)}{\|\tanh f^M(x_i,u) - \tanh f^M(x_j,u)\|}}. \tag{4.8}$$

For maximizing the multistability of an RNN for a given dataset of input sequences, we thus propose to maximize by SGD the VAA^* of batches of hidden states resulting from different input sequences, at random time steps. For each batch of hidden states, a constant input perturbation is randomly sampled from $u \sim \mathcal{N}(0,1)$ in order to stabilize the RNN hidden states over M time steps. However, as can be seen from equation (4.8), maximising the VAA^* only occurs when all hidden states are infinitely distant, which is not desirable for learning efficiently. In practice, we thus use SGD to get the VAA^* of each layer as close as possible to $k = 0.95$, as this proved empirically to maximize the number of attractors (see Figure 4.10) while avoiding too extreme states that could arise from the approximation of the VAA with C^*. In Subappendix 4.D.3, we show on the copy first input benchmark with $T \in \{50, 300, 600\}$ that the warmup

procedure improves learning for a wide range of k. It shows the robustness of our findings with respect to some hyperparameter variation. The loss used is thus given by,

$$\mathcal{L}(v, k) = \frac{1}{L} \sum_{i=1}^{L} (v_i - k)^2, \tag{4.9}$$

where $v_i = \text{VAA}^*_{M,\varepsilon}(f^i, \mathcal{X}, u)$ is the estimated multistability of layer i and L is the number of layers in the RNN. Maximizing the VAA^* of each layer separately allows one to tackle the problem of layer convergence as identified above. To avoid over-fitting problems, M is sampled uniformly in $\{1, \ldots, M_{\max}(s)\}$ at gradient step s, where $M_{\max}(s) = \min(M^*, 1 + c \cdot s)$ with M^* the maximum stabilization period and c the stabilization period increment. This progressive increase is required for reaching multistability smoothly, avoiding gradients problems. For the supervised learning tasks, the batches of input sequences are sampled in the training set. For the reinforcement learning tasks, batches of input sequences are sampled from the exploration policy. Algorithm 4.3 details the whole warmup procedure for a dataset $\mathcal{D}$ of input sequences.

Algorithm 4.3: Recurrent neural network warmup.

parameters: $S \in \mathbb{N}$ the number of gradient steps,
$n \in \mathbb{N}$ the batch size,
$\alpha \in \mathbb{R}^+$ the learning rate,
$k \in [0, 1]$ the target average $\text{VAA}^*_{M,\varepsilon}$,
$M^* \in \mathbb{N}$ the maximum stabilization period,
$c \in \mathbb{N}$ the stabilization period increment,
$\varepsilon \in \mathbb{R}^+$ tolerance when considering state similarity,
L the number of layers in the RNN.

inputs: $\mathcal{D} = \{u^1_{1:T_1}, \ldots, u^N_{1:T_N}\}$ a training set of N input sequences,
$\theta \in \mathbb{R}^{d_\theta}$ the parameters of the network.

for $s = 1, \ldots, S$ **do**
- Sample a batch $\mathcal{B}$ of n sequences in $\mathcal{D}$ without replacement $\mathcal{B} \sim \mathcal{U}^n(\mathcal{D})$.
- Sample a random hidden state in each sequence $\mathcal{X} = \text{RandomHiddenStates}(\mathcal{B}, \theta)$.
- Sample $M \sim \mathcal{U}(\{1, \ldots, \min(M, 1 + s \cdot c)\})$.
- **for** $i = 1, \ldots, L$ **do**
  - Sample $u \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$.
  - Set $v_i = \text{VAA}^*_{M,\varepsilon}(f^i, \mathcal{X}, u)$ where f^i is the update function of the i^{th} RNN layer.
- Compute loss $L = \mathcal{L}(v, k)$ where $v = \begin{pmatrix} v_1 & \cdots & v_L \end{pmatrix}$.
- Compute gradient $g = \nabla_\theta L$ with BPTT (over stabilization period and input sequence).
- Update parameters $\theta = \theta - \alpha g$.
- Update maximum stabilization period $M^* = M^* + c$.

We show in Figure 4.10 that the warmup procedure effectively increases the VAA^* of each layer in an RNN. Furthermore, we can also see on the right in Figure 4.10 that as the warmup procedure is carried out, the true VAA measure of the RNN is increasing as well, even reaching 1 as the warmup procedure ends.

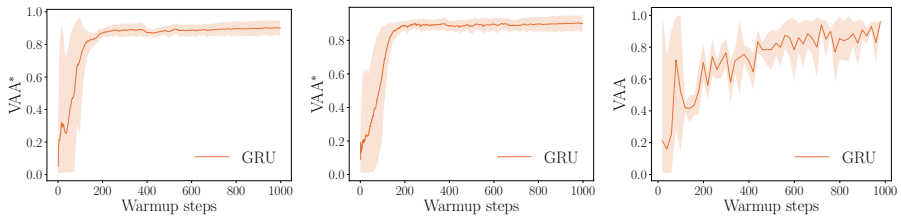


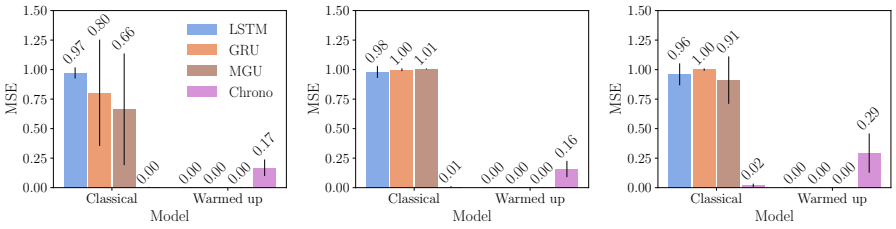
Figure 4.10: Evolution of the VAA* for a two-layer GRU (left and middle) and of the VAA of the network (right) during warmup. This network is warmed up on the denoising dataset and results were averaged over three runs.

4.6.2 Experiments

To demonstrate the impact of warming up RNNs on information restitution tasks, sequence classification tasks, and in partially observable RL environment, we tackle all benchmarks introduced in Section 4.4. We train the LSTM, GRU and MGU cells with and without warmup and show that their performance is greatly improved with warmup. As chrono-initialized LSTMs are known to work well, we also compare our results to such cells, with and without warmup. The hyperparameters were chosen to the same values as in previous section. The goal here is not to measure the best performance of each architecture with or without warmup but rather to measure, for a given architecture and optimization procedure with fixed hyperparameters, whether the warmup initialization procedure provides a better learning for different benchmarks. In Subappendix 4.D.2, we show that those results also hold for other hyperparameters for the permuted row sequential MNIST benchmark. In addition, in the Subsection 4.6.4, we compare the performance of all cells with and without warmup with optimized hyperparameters. All averages and standard deviations reported were computed over three different training sessions. The optimal parameters for warming up can vary depending on architectures and needs, but we found $\alpha = 1e^{-2}$, $c = 10$, $S = 100$, $n = 200$ and $M^* = 200$ to be a good choice.

Copy first input benchmark As can be seen from Figure 4.11, warming up RNNs greatly improves performances in the copy first input benchmark, for any sequence length $T \in \{50, 300, 600\}$. Indeed, classically initialized RNNs have an average loss above 0.500 after 50 epochs, while all warmed-up RNNs have an average loss below 0.001 after 50 epochs. On the other hand, the chrono-initialized LSTM performs better when it is not warmed up. Even if the chrono-initialization and the warmup both promote the learning of long-term dependencies, combining them seems to have the opposite effect, leading to less performant model.

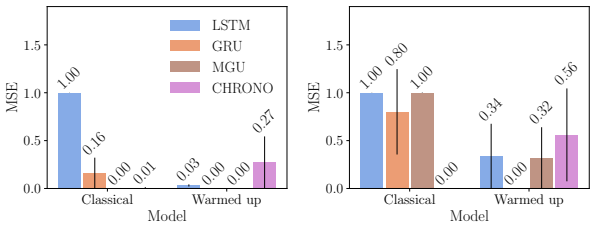
Denoising benchmark As far as the denoising benchmark is concerned, Figure 4.12 shows that warmed-up cells always perform better than classically initialized ones, on sequences of length $T = 200$. However, it can be noted that the average loss is still quite significant after 50 epochs for the LSTM and MGU cells, in the case of a forgetting period of $N = 100$. As for the copy first input benchmark, the chrono-initialized cells perform worse when warmed-up which



(a) $T = 50$ (b) $T = 300$ (c) $T = 600$

Figure 4.11: Test MSE loss for the copy first input benchmark with different sequence lengths T. Mean and standard deviation are reported after 50 epochs.

suggests once again that the chrono initialization interacts disadvantageously with the warmup procedure.



(a) $N = 5$ (b) $N = 100$

Figure 4.12: Test MSE loss for the denoising benchmark with different forgetting periods N and $T = 200$. Mean and standard deviation are reported after 50 epochs.

T-Maze benchmark On the left in Figure 4.13, we can see the evolution of the expected cumulative reward of the DRQN policy for the T-Maze environment as a function of the number of episodes of interaction. It is more than clear that all warmed-up cells and bistable cells (i.e., BRC and NBRC), are better than the classically initialized ones on this RL benchmark. As for the other benchmarks, the chrono-initialized LSTMs seem to interact disadvantageously with the warmup procedure. In any case, it can be noted that the chrono-initialized LSTMs are always among the worse cells for this benchmark, with and without warmup. Furthermore, we can see that warming up cells improves their performance even more as the length of the T-Maze increases, suggesting that the warmup procedure and the multistability of an RNN indeed help to tackle tasks with long time dependencies. On the right in Figure 4.13, we can see the number of episodes required to reach the optimal policy for each cell. It is clear that warming up a cell speeds up the convergence towards the optimal policy when time dependencies become large. Indeed, for $L = 200$, all warmed-up cells reach the optimal policy before any classically initialized cell, except for the chrono-initialized LSTM.

Permuted sequential MNIST In Figure 4.14, we can see the test accuracies after 70 epochs on the permuted sequential MNIST benchmark. It is clear that

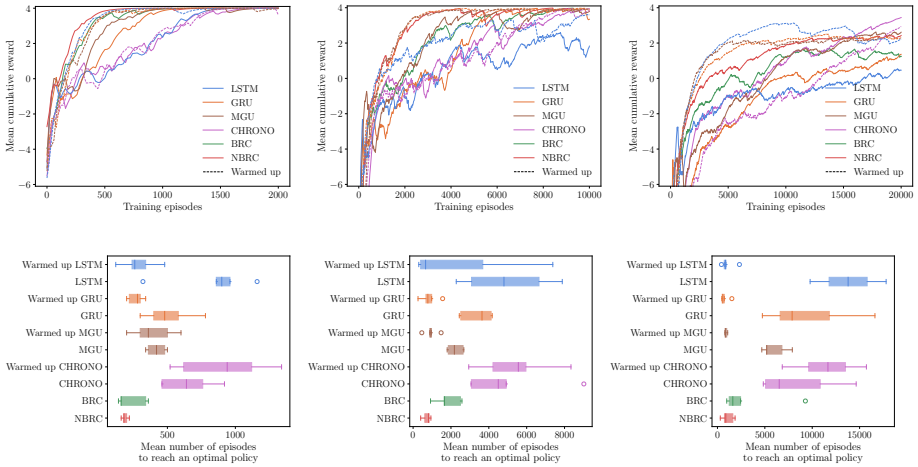


Figure 4.13: Evolution of the mean cumulative reward obtained by warmed-up and classic agents during their training (up) and mean number of episodes required to reach the optimal policy (down) on T-Mazes of length 20 (left), 100 (center) and 200 (right).

the warmup initialization does not help in this task. For the LSTM and GRU, the warmed-up cells are even worse than the classic cells. This confirms that some tasks such as this sequence classification benchmark needs more transient dynamics instead of multistable ones.

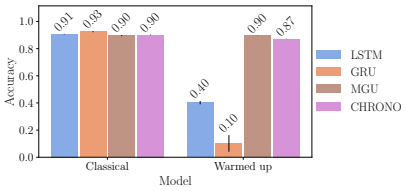
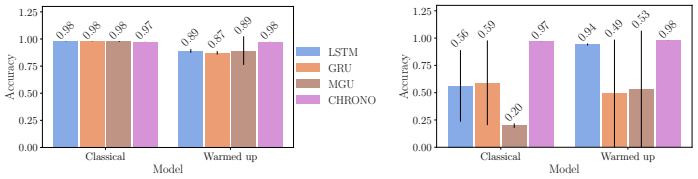


Figure 4.14: Test accuracy for the permuted sequential MNIST benchmark. Mean and standard deviation are reported after 70 epochs.

Permuted line-sequential MNIST In Figure 4.15, we can see the accuracies of each cell after 70 epochs on the test set of the permuted line-sequential MNIST benchmark. For a sequence length of 100 (i.e., $N = 72$), it is clear that the classically initialized cells are better at this task. As for the permuted sequential MNIST, this shows that transient dynamics are important for those sequence classification tasks, as opposed to information restitution tasks.

4.6.3 Recurrent Double-Layers

As shown in the previous section and mentioned in the literature [Sussillo and Barak, 2013], the importance of the transient dynamics of RNNs should not be neglected for prediction. Indeed, it is easy to see why transient dynamics can be of importance when trying to tackle a regression task. If information is only stored in the form of attractors, then there can only be a limited number



(a) $N = 72$ (b) $N = 472$

Figure 4.15: Test accuracy for the permuted line-sequential MNIST benchmark for different forgetting periods N. Mean and standard deviation are reported after 70 epochs. We note that when N equals 72 (472) the resulting image has 100 (500) lines.

of states the network can take, making it very hard to get precise predictions. We observe that when warming up neural networks they tend to lose predictive accuracy, at the benefit of easier training on longer sequences. This leads one to think that RNNs should be built to have both rich transient and multistable dynamics. We thus propose using a double-layer architecture that allows one to get precise predictions while maintaining the benefits of warmup. We simply split each recurrent layer in two equal parts and only warmup one of them. In this double architecture, the hidden states sizes are divided by two compared to the simple architecture, for a fairer comparison. This allows to endow some part of each layer with multistability, while the other remains monostable with richer transient dynamics. A double-layer structure is depicted in Figure 4.16.

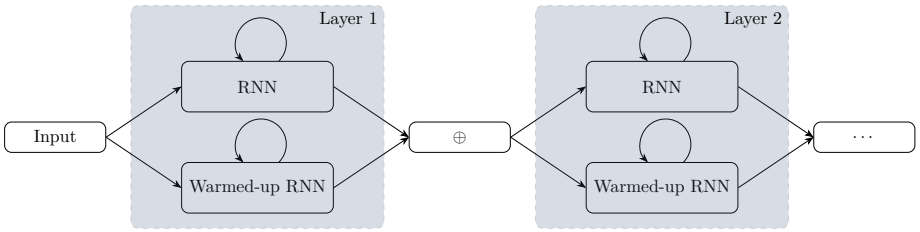
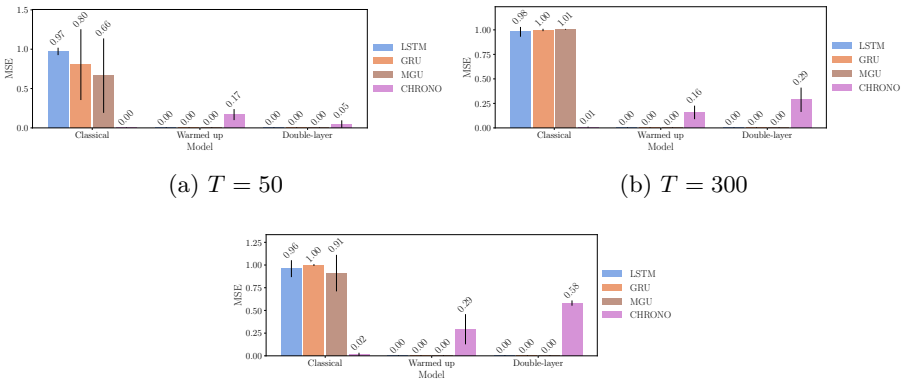


Figure 4.16: Double-layer architecture.

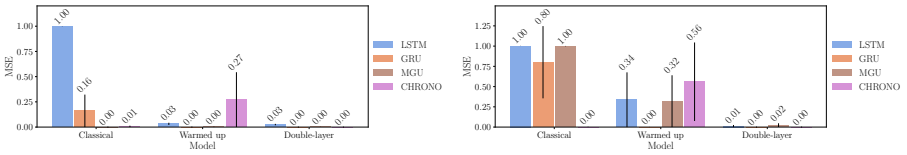
As can be seen from Figure 4.17, Figure 4.18, Figure 4.19 and Figure 4.20, the double-layer architecture is always among the best performing architecture, for all four supervised learning benchmarks and for the LSTM, GRU and MGU cells, when using the same standard hyperparameters of the previous sections. Even the chrono-initialized LSTMs perform well with the double-layer architecture except on the copy first input benchmark. It shows that the double-layer architecture combines both the transient and multistable features of an RNN. In addition, we can see in Figure 4.20 that the double-layer architecture is significantly better than the other architecture, for all types of cell, on the permuted line-sequential MNIST benchmark with a forgetting length of $N = 472$, a problem that requires both transient and multistable dynamics. In addition, we show in Appendix 4.F that the double-layer architectures without partial warmup generally perform worse that the classic architectures. This ensures



(a) $T = 50$ (b) $T = 300$

(c) $T = 600$

Figure 4.17: Test MSE loss for the copy first input benchmark with different sequence lengths T. Mean and standard deviation are reported after 50 epochs.



(a) $N = 5$ (b) $N = 100$

Figure 4.18: Test MSE loss for the denoising benchmark with different forgetting periods N and $T = 200$. Mean and standard deviation are reported after 50 epochs.

that the partial warmup is the most important factor for the performance of the double-layer architecture. In Figure 4.21, we can visualize the evolution of the validation loss averaged over 5 training sessions on the denoising benchmark for the LSTM, GRU and MGU cells, with the three architectures (i.e., classic, warmed up and double). It is clear that the warmed-up and double-layer architectures are better. Additionally, we can see that the double-layer architecture is significantly faster at learning this task for the GRU and MGU cells.

4.6.4 Hyperparameter Optimization

In this section, we study the performance of the different cells in their different versions (i.e., classic, warmed up and double), when the hyperparameters are optimized. In Section 4.6 and Appendix 4.D, we have shown that the warmup procedure and double-layer architecture provides a nice improvement in performance for a wide range of hyperparameters. Here, we consider a more practical setting in which the hyperparameters of a considered cell version can be optimized according to the learning set. We consider a standard hyperparameter selection procedure where the hyperparameters are selected according to the loss on a selection set, averaged over 5 training sessions (see Appendix 4.E for details). Those hyperparameters are then selected for 5 training sessions according to the standard procedure, and the average loss on the test set is reported.

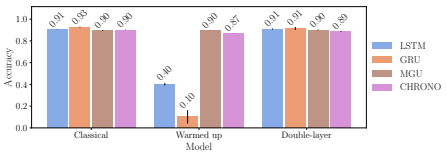


Figure 4.19: Test accuracy for the permuted sequential MNIST benchmark. Mean and standard deviation are reported after 70 epochs.

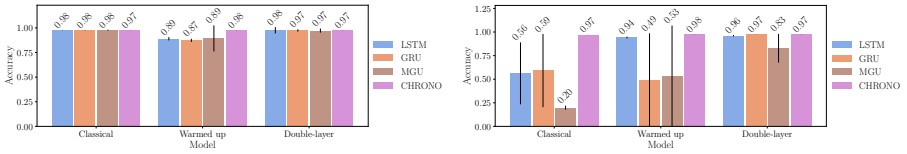
Due to the computational cost of such an optimization procedure, we only consider the most challenging benchmarks of each category, that is the denoising benchmark with $N = 100$ and the permuted line-sequential MNIST benchmark with $N = 472$.

The best hyperparameters are reported in Appendix 4.E for both benchmarks. The test losses obtained using those hyperparameters are given in Figure 4.22. As can be seen by putting Figure 4.22a in perspective with Figure 4.18b, the hyperparameter selection allows all cell versions to reach a lower test MSE for the denoising benchmark. Similarly, by putting Figure 4.22b in perspective with Figure 4.20b, it can be seen that all cell versions reach a higher test accuracy for the MNIST benchmark, when the hyperparameters have been optimized. Figure 4.23 shows the evolution of the validation losses throughout the training procedure for the best hyperparameters of each cell version, averaged over the 5 training sessions, for the denoising benchmark with $N = 100$. It can be seen that the warmup procedure and the double cell architecture still provide a significant advantage in term of convergence speed and final performance. Figure 4.24 shows the evolution of the validation losses throughout the training procedure using the best hyperparameters, for the line-sequential MNIST benchmark with $N = 472$. As for the denoising benchmark, the warmup and the double layer architecture still provide a very significant improvement in term of convergence speed.

4.7 Conclusion

In this work, we introduced a new initialization procedure, called warmup, that improve the ability of recurrent neural networks to learn long time dependencies. This procedure is motivated by recent work that showed the importance of fixed points and attractors for the prediction process of trained RNNs. More precisely, we introduced a lightweight measure called VAA, that can be optimized at initialization in few gradient steps to endow RNNs with multistable dynamics. Warmup can be used with any type of recurrent cell and we show that it vastly improves their performance on problems with long time dependencies. In addition, we introduced a new architecture that combines transient and multistable dynamics through partial warmup. This architecture was shown to reach a better performance than both classic and warmed-up cells on several tasks, including information restitution and sequence classifications tasks.

This work also motivates several future works. First, it can be noted that the double-layer architecture might be worth exploring with different types of cell. We showed here that there are benefits of using different types of initialization for the same type of cell. This might hint at the possibility of having similar



(a) $N = 72$ (b) $N = 472$

Figure 4.20: Test accuracy for the permuted line-sequential MNIST benchmark for different forgetting periods N. Mean and standard deviation are reported after 70 epochs. We note that when N equals 72 (472) the resulting image has 100 (500) lines.

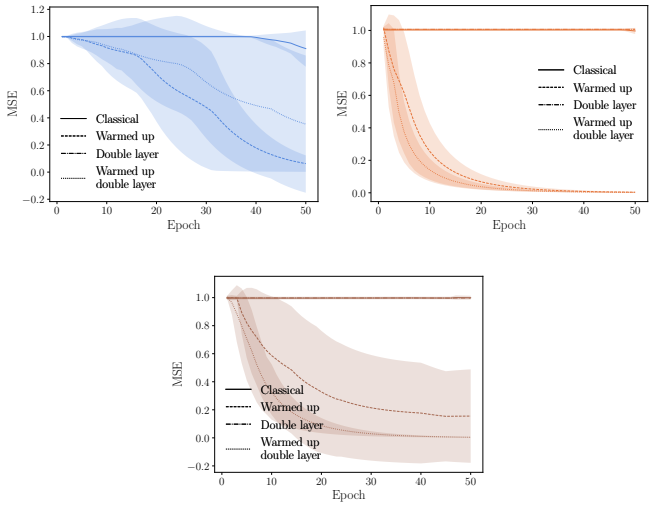
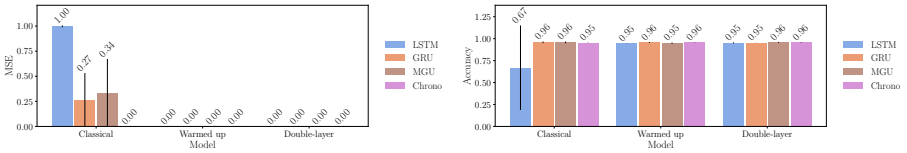


Figure 4.21: Evolution of the validation loss on the denoising benchmark for LSTM, GRU and MGU networks, with $N = 100$ and $T = 200$. For each cell, four versions are considered: the classical one, the warmed-up one and the double-layer one, with and without partial warmup.



(a) Denoising benchmark with $T = 200$ and $N = 100$. (b) Permuted line-sequential MNIST with $N = 472$.

Figure 4.22: Test accuracy for the denoising benchmark and the permuted line-sequential MNIST benchmark with hyperparameter selection on the learning set. Mean and standard deviation are reported after 50 epochs.

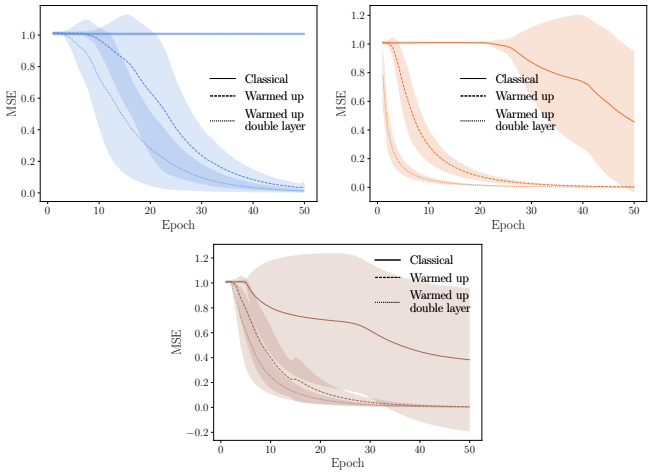


Figure 4.23: Evolution of the validation loss on the denoising benchmark for LSTM, GRU and MGU networks, with $N = 100$ and $T = 200$. For each cell, three versions are considered: the classical one, the warmed-up one and the double-layer one with partial warmup. The hyperparameters of each cell version were optimized on the learning set.

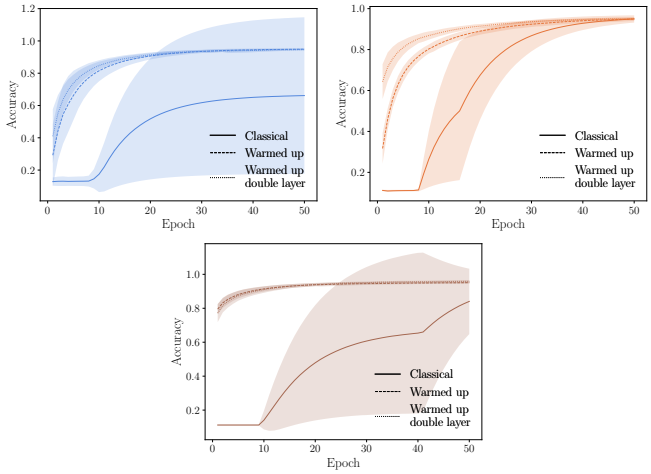


Figure 4.24: Evolution of the validation loss on the line-sequential MNIST benchmark for LSTM, GRU and MGU networks, with $N = 100$ and $T = 200$. For each cell, three versions are considered: the classical one, the warmed-up one and the double-layer one with partial warmup. The hyperparameters of each cell version were optimized on the learning set.

benefits when combining different types of cell that have different dynamical properties in a single recurrent neural network. Furthermore, in this paper we have aimed at maximizing the number of attractors through warmup before training. We noticed however that in some rare cases, networks loose multistability properties when training. Using VAA as a regularization loss to avoid this could be interesting. For online reinforcement learning too, a regularization loss throughout the learning procedure might make more sense than warming up a priori on random trajectories. Moreover, we note that not all benchmarks would benefit from warming up. In fact, it is likely that for several benchmarks, having only a few attractors could be better. In this regard, it would be interesting to try to warm up in order to reach a specific number of attractors, rather than for maximizing them. Finally, the warmup procedure maximizes reachable multistability for a particular dataset of input sequences. Warming up on totally random input sequences would result in a simpler procedure that might still provide a good initialization for reaching multistability.

This work also present some limitations. First, the VAA is not discriminating limit cycles from fixed point attractors. In addition, states that are on the same limit cycles but far from each others are not considered in the basin. Moreover, the warmup maximizes the number of attractors present in all hidden states, while we might want the hidden states from a same input sequence to belong to a single basin of attraction. Finally, the stability of the RNN is measured for a stable input, an assumption that is unrealistic in our experiments and in general. It might be worth exploring those problems in future works.

4.A Recurrent Neural Network Architectures

Formally, an RNN architecture is defined by its update function f, its output function g and its initialization function h that are parameterized by a vector $\theta \in \mathbb{R}^d$. Given a sequence of inputs $u_{1:T} = [u_1, \dots, u_T]$, with $T \in \mathbb{N}$ and $u_t \in \mathbb{R}^n$, the RNN maintains a hidden state x_t and output a prediction o_t according to,

$$x_t = f(x_{t-1}, u_t; \theta),\ t = 1, \dots, T, \tag{4.10}$$

$$o_t = g(x_t; \theta),\ t = 1, \dots, T, \tag{4.11}$$

$$x_0 = h(\theta). \tag{4.12}$$

RNNs can be composed of L layers that are linked sequentially through $u_t^i = o_t^{i-1}$ with $u_t^1 = u_t$ and $o_t = o_t^L$, where o_t^i denotes the output of layer i and u_t^i its input. In this case, each layer i has its own update function f^i, output function g^i and initialization function h^i.

In the following, we give the update function f and output function g of a single layer for each architecture considered in this work. As far as the initial hidden state is concerned, it is always chosen to zero, i.e., $h(\theta) = 0$. Note that $\sigma(x) = \frac{1}{1+e^{-x}}$ denote the sigmoid activation function, and $\odot$ to denote the Hadamard product.

Long short-term memory. The LSTM update and output functions are defined from the following intermediate values,

$$f_t = \sigma(W_{fu} u_t + W_{fh} h_{t-1} + b_t), \tag{4.13}$$

$$i_t = \sigma(W_{iu} u_t + W_{ih} h_{t-1} + b_i), \tag{4.14}$$

$$r_t = \sigma(W_{ou} u_t + W_{oh} h_{t-1} + b_r), \tag{4.15}$$

$$\tilde{c}_t = \tanh(W_{cu} u_t + W_{ch} h_{t-1} + b_c), \tag{4.16}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{4.17}$$

$$h_t = r_t \odot \tanh(c_t). \tag{4.18}$$

The hidden state is given by $x_t = f(x_{t-1}, u_t; \theta) = [h_t, c_t]$, and the output is given by $o_t = g(x_t; \theta) = h_t$. The parameters of the LSTM network are $\theta = (W_{fu}, W_{fh}, W_{iu}, W_{ih}, W_{ou}, W_{oh}, W_{cu}, W_{ch}, b_t, b_i, b_r, b_c)$.

Gated recurrent unit. The GRU update and output functions are defined from the following intermediate values,

$$z_t = \sigma(W_{zu} u_t + W_{zh} h_{t-1} + b_z), \tag{4.19}$$

$$r_t = \sigma(W_{ru} u_t + W_{rh} h_{t-1} + b_r), \tag{4.20}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tanh(W_{hu} u_t + r_t \odot W_{hh} h_{t-1} + b_h). \tag{4.21}$$

The hidden state is given by $x_t = f(x_{t-1}, u_t; \theta) = h_t$, and the output is given by $o_t = g(x_t; \theta) = h_t$. The parameters of the GRU network are $\theta = (W_{zu}, W_{zh}, W_{ru}, W_{rh}, W_{hu}, W_{hh}, b_z, b_r, b_h)$.

Bistable recurrent cell. The BRC update and output functions are defined from the following intermediate values,

$$c_t = \sigma(W_{cu} u_t + w_c \odot h_{t-1} + b_c), \tag{4.22}$$

$$a_t = 1 + \tanh(W_{au} u_t + w_a \odot h_{t-1} + b_a), \tag{4.23}$$
$$h_t = c_t \odot h_{t-1} + (1 - c_t) \odot \tanh(W_{hu} + a_t \odot h_{t-1} + b_h). \tag{4.24}$$

The hidden state is given by $x_t = f(x_{t-1}, u_t; \theta) = h_t$, and the output is given by $o_t = g(x_t; \theta) = h_t$. The parameters of the BRC network are $\theta = (W_{cu}, w_c, W_{au}, w_a, W_{hu}, b_c, b_a, b_h)$.

Neuromodulated bistable recurrent cell. The NBRC update and output functions are defined from the following intermediate values,

$$c_t = \sigma(W_{cu} u_t + W_{ch} h_{t-1} + b_c), \tag{4.25}$$
$$a_t = 1 + \tanh(W_{au} u_t + W_{ah} h_{t-1} + b_a), \tag{4.26}$$
$$h_t = c_t \odot h_{t-1} + (1 - c_t) \odot \tanh(W_{hu} + a_t \odot h_{t-1} + b_h). \tag{4.27}$$

The hidden state is given by $x_t = f(x_{t-1}, u_t; \theta) = h_t$, and the output is given by $o_t = g(x_t; \theta) = h_t$. The parameters of the NBRC network are $\theta = (W_{cu}, W_{ch}, W_{au}, W_{ah}, W_{hu}, b_c, b_a, b_h)$.

Minimal gated unit. The MGU update and output functions are defined from the following intermediate values,

$$f_t = \sigma(W_{fu} u_t + W_{fh} h_{t-1} + b_f), \tag{4.28}$$
$$\tilde{h}_t = \tanh(W_{hu} u_t + W_{hh}(f_t \odot h_{t-1}) + b_h), \tag{4.29}$$
$$h_t = f_t \odot \tilde{h} + (1 - f_t) \odot h_{t-1}. \tag{4.30}$$

The hidden state is given by $x_t = f(x_{t-1}, u_t; \theta) = h_t$, and the output is given by $o_t = g(x_t; \theta) = h_t$. The parameters of the MGU network are $\theta = (W_{fu}, W_{fh}, W_{hu}, W_{hh}, b_f, b_h)$.

4.B Partially Observable Markov Decision Processes

Formally, a POMDP P is a tuple $P = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, P, \gamma)$ where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, and $\mathcal{O}$ is the observation space. The initial state distribution P gives the probability $P(s_0)$ of $s_0 \in \mathcal{S}$ being the initial state of the decision process. The dynamics are described by the transition distribution T that gives the probability $T(s_{t+1} \mid s_t, a_t)$ of $s_{t+1} \in \mathcal{S}$ being the state resulting from action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$. The reward function R gives the immediate reward $r_t = R(s_t, a_t, s_{t+1})$ obtained after each transition. The observation distribution O gives the probability $O(o_t \mid s_t)$ to get observation $o_t \in \mathcal{O}$ in state $s_t \in \mathcal{S}$. Finally, the discount factor $\gamma \in [0, 1[$ weights the relative importance of future rewards.

Taking a sequence of t actions $(a_{0:t-1})$ in the POMDP conditions its execution and provides a sequence of $t+1$ observations $(o_{0:t})$. Together, they compose the history $h_t = (o_{0:t}, a_{0:t-1}) \in \mathcal{H}_t$ until time step t, where $\mathcal{H}_t$ is the set of such histories. Let $h \in \mathcal{H}$ denote a history of arbitrary length sampled in the POMDP, and let $\mathcal{H} = \bigcup_{t=0}^{\infty} \mathcal{H}_t$ denote the set of histories of arbitrary length.

A policy $\eta \in H$ in a POMDP is a mapping from histories to actions, where $H = \mathcal{H} \to \mathcal{A}$ is the set of such mappings. A policy $\eta^* \in H$ is said to be optimal when it maximizes the expected discounted sum of future rewards starting from any history $h \in \mathcal{H}$,

$$\eta^* \in \arg\max_{\eta \in H} \mathbb{E}^{\eta} \left[\sum_{t=0}^{\infty} \gamma^t R_t \;\middle|\; H_0 = h \right], \; \forall h \in \mathcal{H}. \tag{4.31}$$

The history-action value function, or Q-function, is defined as the maximal expected discounted reward that can be gathered, starting from a history $h \in \mathcal{H}$ and an action $a \in \mathcal{A}$,

$$Q(h, a) = \max_{\eta \in H} \mathbb{E}^{\eta} \left[\sum_{t=0}^{\infty} \gamma^t R_t \;\middle|\; H_0 = h, A_0 = a \right], \tag{4.32}$$

The Q-function is also the unique solution of the Bellman equation [Smallwood and Sondik, 1973, Kaelbling et al., 1998, Porta et al., 2006],

$$Q(h, a) = \mathbb{E}\left[R + \gamma \max_{a' \in \mathcal{A}} Q(H', A') \;\middle|\; H = h, A = a \right], \; \forall h \in \mathcal{H}, \; \forall a \in \mathcal{A}, \tag{4.33}$$

where $H' = (H, A, O')$ and R is the immediate reward obtained when taking action A in history R. From (4.31) and (4.32), it can be noticed that any optimal policy satisfies,

$$\eta^*(h) \in \arg\max_{a \in \mathcal{A}} Q(h, a), \; \forall h \in \mathcal{H}. \tag{4.34}$$

4.C Deep Recurrent Q-learning

The DRQN [Hausknecht and Stone, 2015] algorithm aims at learning a parametric approximation Q_θ of the Q-function, where $\theta \in \mathbb{R}^{d_\theta}$ is the parameter vector of a recurrent neural network. This algorithm is motivated by equation (4.34) that shows that an optimal policy can be derived from the Q-function. The strategy consists of minimising with respect to θ, for all (h, a), the distance between the estimation $Q_\theta(h, a)$ of the LHS of equation (4.33), and the estimation of the expectation $\mathbb{E}[r + \gamma \max_{a' \in \mathcal{A}} Q_\theta(h', a')]$ of the RHS of equation (4.33). This is done by using transitions (h, a, r, o', h') sampled in the POMDP, with $h' = (h, a, o')$.

In practice, this algorithm interleaves the generation of episodes and the update of the estimation Q_θ. Indeed, in the DRQN algorithm, the episodes are generated with the ε-greedy policy derived from the current estimation Q_θ. This stochastic policy selects actions according to $\arg\max_{a \in \mathcal{A}} Q_\theta(\cdot, a)$ with probability $1 - \varepsilon$, and according to an exploration policy with probability ε. This exploration policy is defined by a probability distribution $\mathcal{E}(\mathcal{A}) \in \mathcal{P}(\mathcal{A})$ over the actions, where $\mathcal{P}(\mathcal{A})$ is the set of probability measures over the action space $\mathcal{A}$. The DRQN algorithm also introduces a truncation horizon H such that the histories generated in the POMDP have a maximum length of H. Moreover, a replay buffer of histories is used and the gradient is evaluated on a batch of histories sampled from this buffer. Furthermore, the parameters θ are updated with the Adam algorithm [Kingma and Ba, 2014]. Finally, the target

$r_t + \gamma \max_{a \in \mathcal{A}} Q_{\theta'}(h_{t+1}, a)$ is computed using a past version $Q_{\theta'}$ of the estimation Q_θ with parameters θ' that are updated to θ less frequently, which eases the convergence towards the target, and ultimately towards the Q-function.

Algorithm 4.4: Deep recurrent Q-learning.

parameters: $N \in \mathbb{N}$ the buffer capacity,
$C \in \mathbb{N}$ the target update period in term of episodes,
$E \in \mathbb{N}$ the number of episodes,
$H \in \mathbb{N}$ the truncation horizon,
$I \in \mathbb{N}$ the number of gradient steps after each episode,
$\varepsilon \in \mathbb{R}$ the exploration rate,
$\mathcal{E}(\mathcal{A}) \in \mathcal{P}(\mathcal{A})$ the exploration policy probability distribution,
$\alpha \in \mathbb{R}$ the learning rate,
$B \in \mathbb{N}$ the batch size,
$\theta \in \mathbb{R}^{d_\theta}$ the initial parameters of the network,
$\theta' \in \mathbb{R}^{d_\theta}$ the initial parameters of the target network.

Initialize weights θ randomly.
Fill replay buffer $\mathcal{B}$ with transitions from the exploration policy $\mathcal{E}(\mathcal{A})$.
if *warmup* **then**
  Let $\mathcal{D}$ be the set of histories h (input sequences) in replay buffer $\mathcal{B}$.
  Warmup the parameters of the RNN using Warmup$(\mathcal{D}, \theta)$.
for $e = 0, \dots, E-1$ **do**
  if $e \bmod C = 0$ **then**
    Update target network with $\theta' \leftarrow \theta$.
  Draw an initial state s_0 according to P and observe o_0.
  Let $h_0 = (o_0)$.
  for $t = 0, \dots, H-1$ **do**
    Select $a_t \sim \mathcal{E}(\mathcal{A})$ with probability ε, otherwise select $a_t = \arg\max_{a \in \mathcal{A}} \{Q_\theta(h_t, a)\}$.
    Take action a_t and observe r_t and o_{t+1}.
    Let $h_{t+1} = (o_0, a_0, o_1, \dots, o_{t+1})$.
    if $|\mathcal{B}| < N$ **then** add $(h_t, a_t, r_t, o_{t+1}, h_{t+1})$ in replay buffer $\mathcal{B}$
    else replace oldest transition in replay buffer $\mathcal{B}$ by $(h_t, a_t, r_t, o_{t+1}, h_{t+1})$.
    if o_{t+1} *is terminal* **then**
      break
  for $i = 0, \dots, I-1$ **do**
    Sample B transitions $(h_t^b, a_t^b, r_t^b, o_{t+1}^b, h_{t+1}^b)$ uniformly from the replay buffer $\mathcal{B}$.
    Compute targets

$$y^b = \begin{cases} r_t^b + \gamma \max_{a \in \mathcal{A}} \left\{ Q_{\theta'}(h_{t+1}^b, a) \right\} & \text{if } o_{t+1}^b \text{ is not terminal.} \\ r_t^b & \text{otherwise.} \end{cases}$$

    Compute loss $L = \sum_{b=0}^{B-1} \left(y^b - Q_\theta(h_t^b, a_t^b) \right)^2$.
    Compute direction g using Adam optimizer.
    Perform gradient step $\theta \leftarrow \theta + \alpha g$.

The DRQN training procedure is detailed in Algorithm 4.4. In this algorithm, the output of the RNN is $y_t = g(x_t; \theta) \in \mathbb{R}^{|\mathcal{A}|}$, and it gives $Q_\theta(h_t, a)$, $\forall a \in \mathcal{A}$. The hidden states are given by $x_k = f(x_{k-1}, u_k; \theta)$, $\forall k \in \mathbb{N}_0$, with the inputs given by $u_k = (a_{k-1}, o_k)$, $\forall t \in \mathbb{N}$ and $u_0 = (0, o_0)$. From the approximation Q_θ, the policy η_θ is given by $\eta_\theta(h) = \arg\max_{a \in \mathcal{A}} Q_\theta(h, a)$.

In the experiments, the following hyperparameters have been chosen: $N = 8192$,

$C = 20$, $I = 10$, $\varepsilon = 0.2$, $\alpha = 1 \times 10^{-3}$, $B = 32$. The exploration policy and truncation horizon depend on the environment and are thus detailed in the following appendix.

4.D Generalization to Other Hyperparameters

In this section, we study the generalization of the results of this work to other hyperparameters. More precisely, we vary the number of recurrent layers, the number of neurons in each layer, the batch size, and the learning rate. In Subappendix 4.D.1, we study if the VAA increases when learning occurs for the copy first input benchmark with $T = 50$. In Subappendix 4.D.2, we study if the warmup procedure and the double layer architecture improve learning for the permuted row sequential MNIST benchmark with $N = 472$. Finally, in Subappendix 4.D.3, we study the impact of the warmup procedure on the copy first input benchmark with $T = 300$ for different values of k. All averages and standard deviations reported were computed over three different trainings.

4.D.1 Correlation Between Multistability and Learning

In Figure 4.25 and Figure 4.26, we see the evolution of the loss on the validation set and of the VAA for different hyperparameters. There is a clear correlation between learning and multistability, for all choices of hyperparameters. More precisely, it can be seen that learning loss decrease generally starts when the VAA starts increasing. Moreover, the loss is highly correlated with the VAA.

4.D.2 Learning Improvements with the Warmup Procedure

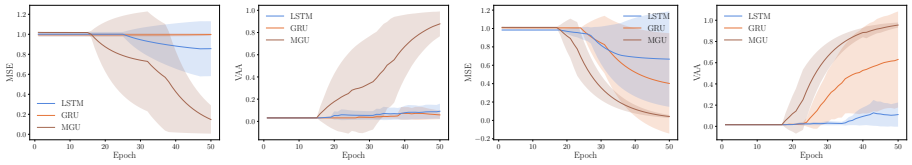
In Figure 4.27 and Figure 4.28, we can see the evolution of the loss on the validation set and the test set accuracy for different hyperparameters. It can be seen from those figures that the warmup procedure and the double layer architecture with partial warmup both improve on the classically initialized GRU architecture. Those improvements are consistent over all hyperparameters choices. It can be noted that the warmup procedure is sometimes better than the double layer architecture in terms of speed of convergence, notably when using a single RNN layer and a small hidden size.

4.D.3 Impact of the Parameter k in the Warmup Procedure

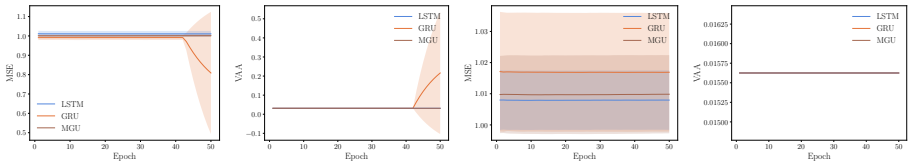
In Figure 4.29, we can see the impact of the target VAA* k used in the warmup procedure on the final test loss, for the copy first input benchmark for different sequence lengths T. It can be seen that for this benchmark with long time dependencies, the higher k, the lower the MSE.

4.E Hyperparameters Optimization

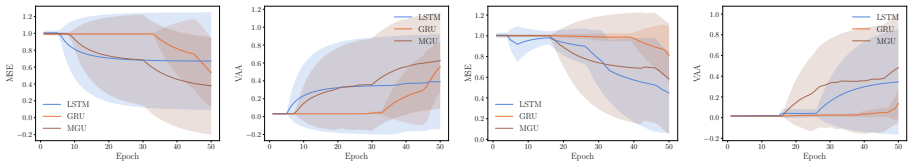
In this section, we report the best hyperparameters obtained for each cell version and the final test loss obtained for those hyperparameters, in Table 4.1 for the



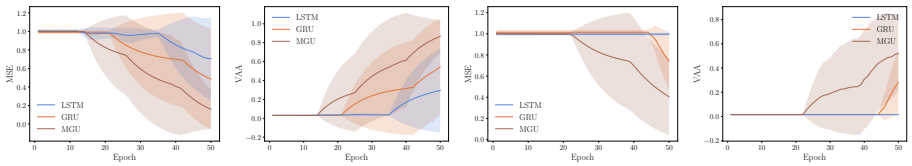
(a) $S = 1, H = 64, \alpha = 1e-3, B = 32$ (b) $S = 1, H = 64, \alpha = 1e-3, B = 64$



(c) $S = 1, H = 64, \alpha = 1e-4, B = 32$ (d) $S = 1, H = 64, \alpha = 1e-4, B = 64$

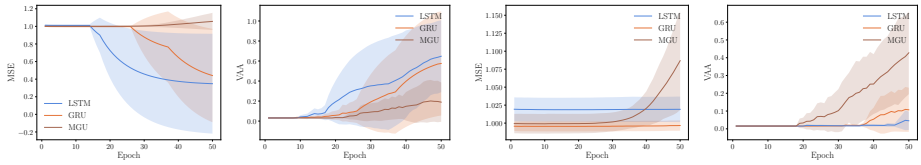


(e) $S = 1, H = 256, \alpha = 1e-3, B = 32$ (f) $S = 1, H = 256, \alpha = 1e-3, B = 64$

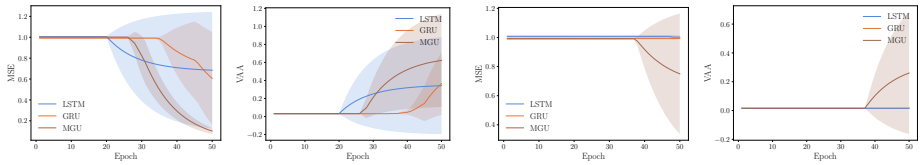


(g) $S = 1, H = 256, \alpha = 1e-4, B = 32$ (h) $S = 1, H = 256, \alpha = 1e-4, B = 64$

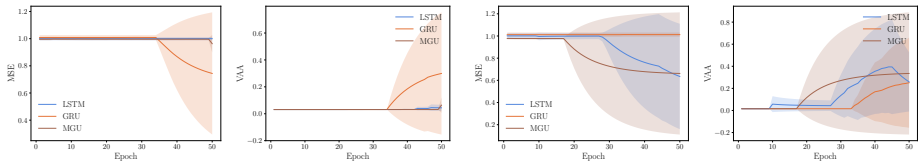
Figure 4.25: Evolution of the validation loss (left) and of the VAA (right) of LSTM, GRU and MGU networks, for the copy first input benchmark.



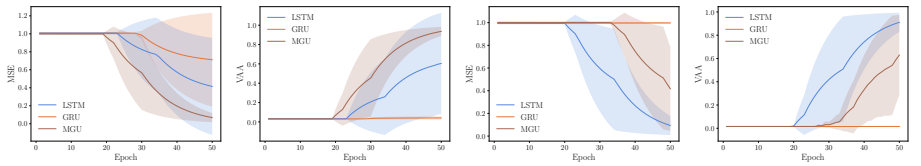
(a) $S = 2, H = 64, \alpha = 1e-3, B = 32$ (b) $S = 2, H = 64, \alpha = 1e-3, B = 64$



(c) $S = 2, H = 64, \alpha = 1e-4, B = 32$ (d) $S = 2, H = 64, \alpha = 1e-4, B = 64$

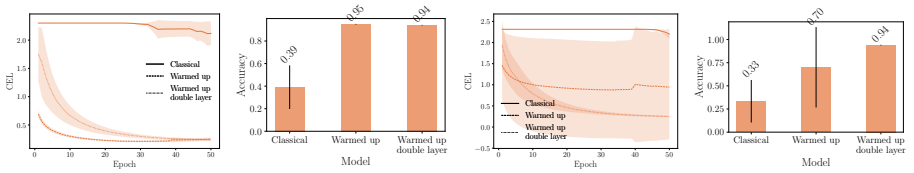


(e) $S = 2, H = 256, \alpha = 1e-3, B = 32$ (f) $S = 2, H = 256, \alpha = 1e-3, B = 64$



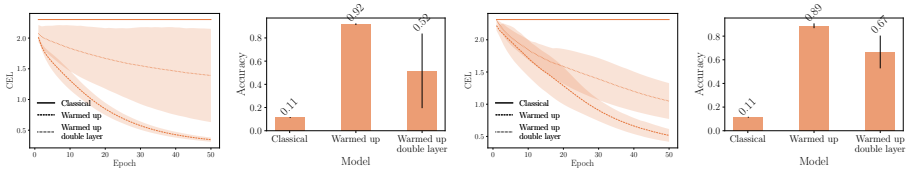
(g) $S = 2, H = 256, \alpha = 1e-4, B = 32$ (h) $S = 2, H = 256, \alpha = 1e-4, B = 64$

Figure 4.26: Evolution of the validation loss (left) and of the VAA (right) of LSTM, GRU and MGU networks, for the copy first input benchmark.



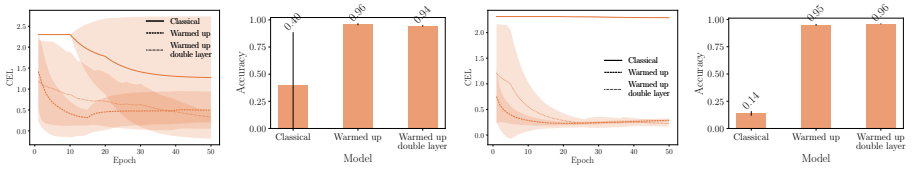
(a) $S = 1, H = 64, \alpha = 1e-3, B = 32$

(b) $S = 1, H = 64, \alpha = 1e-3, B = 64$



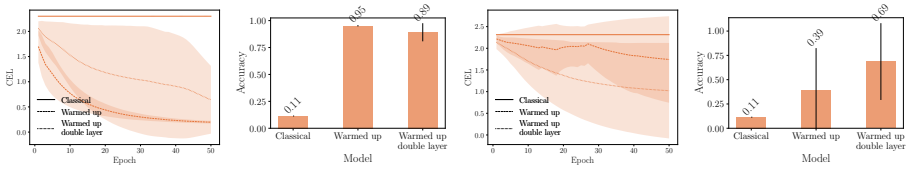
(c) $S = 1, H = 64, \alpha = 1e-4, B = 32$

(d) $S = 1, H = 64, \alpha = 1e-4, B = 64$



(e) $S = 1, H = 256, \alpha = 1e-3, B = 32$

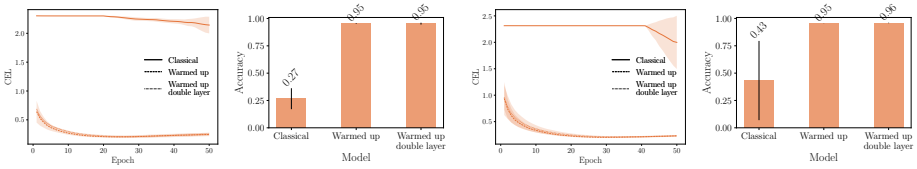
(f) $S = 1, H = 256, \alpha = 1e-3, B = 64$



(g) $S = 1, H = 256, \alpha = 1e-4, B = 32$

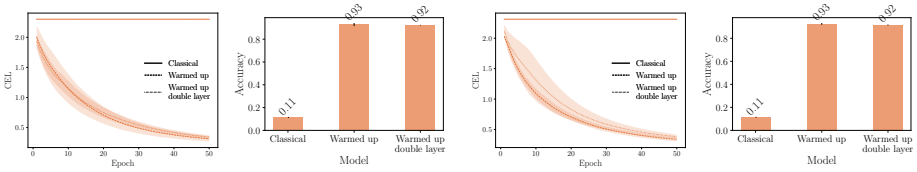
(h) $S = 1, H = 256, \alpha = 1e-4, B = 64$

Figure 4.27: Evolution of the validation loss (left) and test set accuracy after 50 epochs (right) of GRU networks, for the permuted line-sequential MNIST benchmark with $N = 472$.



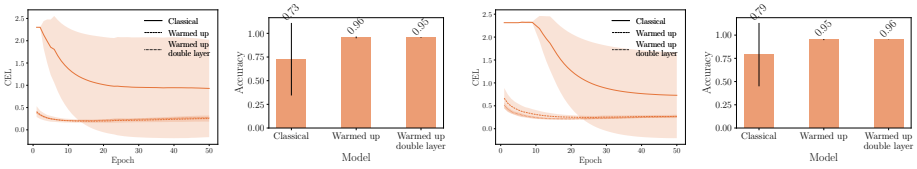
(a) $S = 2, H = 64, \alpha = 1e-3, B = 32$

(b) $S = 2, H = 64, \alpha = 1e-3, B = 64$



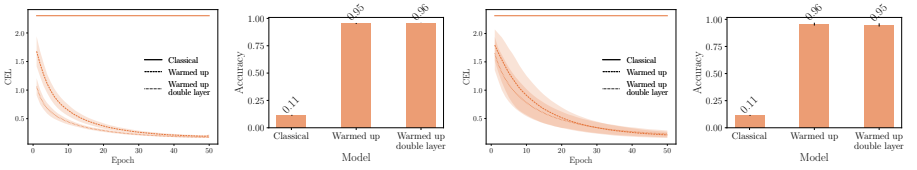
(c) $S = 2, H = 64, \alpha = 1e-4, B = 32$

(d) $S = 2, H = 64, \alpha = 1e-4, B = 64$



(e) $S = 2, H = 256, \alpha = 1e-3, B = 32$

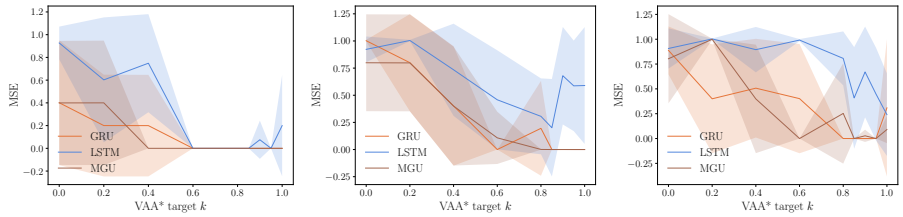
(f) $S = 2, H = 256, \alpha = 1e-3, B = 64$



(g) $S = 2, H = 256, \alpha = 1e-4, B = 32$

(h) $S = 2, H = 256, \alpha = 1e-4, B = 64$

Figure 4.28: Evolution of the validation loss (left) and test set accuracy after 50 epochs (right) of GRU networks, for the permuted line-sequential MNIST benchmark with $N = 472$.



(a) $T = 50$ (b) $T = 300$ (c) $T = 600$

Figure 4.29: Mean squared error ($\pm$ standard deviation) of different architecture for different value of target VAA* k on the copy first input test set for different values of T.

		L	H	α	B	MSE
LSTM	Classical	3	512	1×10^{-3}	32	0.9970 ± 0.0089
	Double	3	256	5×10^{-4}	64	$\mathbf{0.0004 \pm 0.0001}$
	Warmup	3	256	1×10^{-3}	64	0.0010 ± 0.0009
GRU	Classical	1	512	1×10^{-3}	32	0.2656 ± 0.4593
	Double	2	256	1×10^{-3}	32	$\mathbf{0.0002 \pm 0.0001}$
	Warmup	1	512	5×10^{-4}	64	$\mathbf{0.0002 \pm 0.0001}$
MGU	Classical	3	512	1×10^{-3}	32	0.3356 ± 0.5695
	Double	2	256	5×10^{-4}	32	$\mathbf{0.0003 \pm 0.0000}$
	Warmup	1	256	1×10^{-3}	32	$\mathbf{0.0003 \pm 0.0002}$
Chrono	Classical	1	512	1×10^{-3}	32	$\mathbf{0.0003 \pm 0.0002}$
	Double	1	512	1×10^{-3}	32	0.0004 ± 0.0003
	Warmup	1	512	1×10^{-3}	32	0.0004 ± 0.0002
BRC	Classical	3	256	1×10^{-3}	32	$\mathbf{0.0006 \pm 0.0001}$
NBRC	Classical	1	512	1×10^{-3}	32	$\mathbf{0.0001 \pm 0.0000}$

Table 4.1: Test mean squared error after hyperparameter selection in the denoising benchmark.

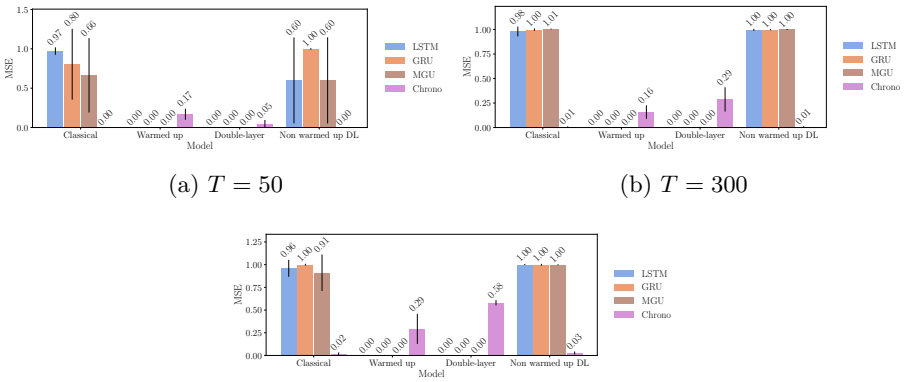
denoising benchmark with $N = 100$ and in Table 4.2 for the line-sequential MNIST benchmark with $N = 472$. The hyperparameter selection procedure is described hereafter. First, the dataset is split into the learning set and the test set. Then, the learning set is split into three sets: the training set, the validation set and the selection set. The network is then trained according to the standard procedure: the final weights are those that have obtained the lowest loss on the validation set, throughout the training on the training set. Those weights are then evaluated on the selection set. This procedure is repeated five times for each set of hyperparameters, with different splits of the learning set each time. Note that those 5 different splits are the same for all cell versions. The set of hyperparameters having obtained the lowest loss on average on the selection set is selected. Using those hyperparameters, the cells are then trained 5 times on the learning set, using a standard training-validation split, and the average score obtained on the test set is reported. The sets of hyperparameters that are considered are given by a grid search.

4.F Double-Layer Architecture without Partial Warmup

In this section, we show the performance of all cells on the copy first input and denoising benchmarks including the double-layer architecture without partial warmup. As can be seen from Figure 4.30 and Figure 4.31 the double-layer architecture without partial warmup generally performs worse than the classic architecture. This ablation study confirms that the partial warmup is the most important factor for the double-layer architecture performance.

		L	H	α	B	Accuracy
LSTM	Classical	2	512	1×10^{-3}	64	0.6693 ± 0.4807
	Double	2	256	5×10^{-4}	32	$\mathbf{0.9519 \pm 0.0058}$
	Warmup	3	256	5×10^{-4}	32	0.9475 ± 0.0008
GRU	Classical	3	512	5×10^{-4}	32	$\mathbf{0.9578 \pm 0.0087}$
	Double	2	512	1×10^{-4}	32	0.9549 ± 0.0011
	Warmup	1	512	1×10^{-4}	32	0.9555 ± 0.0053
MGU	Classical	2	512	5×10^{-4}	64	$\mathbf{0.9576 \pm 0.0085}$
	Double	2	512	5×10^{-4}	64	0.9562 ± 0.0045
	Warmup	2	256	5×10^{-4}	32	0.9485 ± 0.0073
Chrono	Classical	1	256	1×10^{-3}	32	0.9545 ± 0.0020
	Double	1	256	1×10^{-3}	32	$\mathbf{0.9575 \pm 0.0029}$
	Warmup	1	256	1×10^{-3}	32	0.9562 ± 0.0017
BRC	Classical	2	512	5×10^{-4}	32	$\mathbf{0.9589 \pm 0.0064}$
NBRC	Classical	2	512	5×10^{-4}	64	$\mathbf{0.9600 \pm 0.0006}$

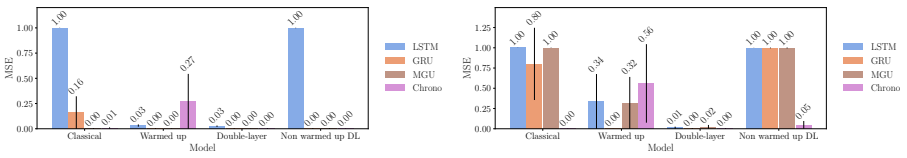
Table 4.2: Test accuracy after hyperparameter selection in the line-sequential MNIST benchmark.



(a) $T = 50$ (b) $T = 300$

(c) $T = 600$

Figure 4.30: Test MSE loss for the copy first input benchmark with different sequence lengths T. Mean and standard deviation are reported after 50 epochs.



(a) $N = 5$ (b) $N = 100$

Figure 4.31: Test MSE loss for the denoising benchmark with different forgetting periods N and $T = 200$. Mean and standard deviation are reported after 50 epochs.

Part II

Leveraging Additional Information

Leveraging Additional Information

In this part, we question the learning paradigm for reinforcement learning in partially observable Markov decision processes. First, we study how eventual additional information about the environment can be leveraged at training time for learning to memorize relevant information. It is achieved by adapting an existing world model to learn predicting future information about the environment instead of future observations, which is shown to learn a sufficient statistic of the history. Second, we question the theoretical foundations of asymmetric reinforcement learning algorithms, by studying why leveraging additional information would improve learning. More precisely, we show that the asymmetric actor-critic algorithm does not suffer from aliasing in the agent state when evaluating the policy, compared to its symmetric counterpart.

Chapter 5

Remembering by Predicting Additional Information

Informed POMDP: Leveraging Additional Information in Model-Based RL. Gaspard Lambrechts, Adrien Bolland and Damien Ernst.

From the paper presented at the *Reinforcement Learning Conference* and published in the *Reinforcement Learning Journal*.

Abstract

In this work, we generalize the problem of learning through interaction in a POMDP by accounting for eventual additional information available at training time. First, we introduce the informed POMDP, a new learning paradigm offering a clear distinction between the information at training and the observation at execution. Next, we propose an objective that leverages this information for learning a sufficient statistic of the history for optimal control. We then adapt this informed objective to learn a world model able to sample latent trajectories. Finally, we empirically show a learning speed improvement in several environments using this informed world model in the Dreamer algorithm. These results and the simplicity of the proposed adaptation advocate for a systematic consideration of eventual additional information when learning in a POMDP using model-based RL.

5.1 Introduction

Reinforcement learning (RL) aims to learn to act optimally through interaction with environments whose dynamics are unknown. A major challenge in this field is partial observability, where only a partial observation o of the Markovian state of the environment s is available for taking action a. Such an environment can be formalized as a partially observable Markov decision process (POMDP). In this context, an optimal policy $\eta(a|h)$ generally depends on the history h of all observations and previous actions, which grows linearly with time. Fortunately, it is theoretically possible to find a statistic $f(h)$ of the history h that is updated recurrently and that summarizes all relevant information to act optimally. Such a statistic is said to be recurrent and sufficient for optimal control. Formally, a statistic $f(h)$ is recurrent when it is updated according to $f(h') = u(f(h), a, o')$ each time an action a is taken and a new observation o' is received, with $h' = (h, a, o')$. And a statistic $f(h)$ is sufficient for optimal control when there exists an optimal policy $\eta(a|h) = g(a|f(h))$.

In view of the existence of recurrent and sufficient statistics, many approaches have relied on learning a recurrent policy $\eta_{\theta,\phi}(a|h) = g_\phi(a|f_\theta(h))$ using a recurrent neural network (RNN) f_θ for the statistic. These policies are simply trained by stochastic gradient ascent of a RL objective using backpropagation through time [Bakker, 2001, Wierstra et al., 2007, Hausknecht and Stone, 2015, Heess et al., 2015, Zhang et al., 2016, Zhu et al., 2017]. In this case, the RNN learns a sufficient statistic $f_\theta(h)$ as it learns an optimal policy [Lambrechts et al., 2022, Hennig et al., 2023]. Although these approaches theoretically allow implicit learning of a sufficient statistic, sufficient statistics can also be learned explicitly. Notably, many works [Igl et al., 2018, Buesing et al., 2018, Guo et al., 2018, Gregor et al., 2019, Han et al., 2019, Guo et al., 2020, Lee et al., 2020, Hafner et al., 2019, 2020] focused on learning a recurrent statistic that encodes the reward and next observation distribution given the action: $p(r, o'|h, a) = p(r, o'|f(h), a)$, a property known as predictive sufficiency [Bernardo and Smith, 2009]. A recurrent and predictive statistic is indeed proven to be sufficient for optimal control [Subramanian et al., 2022]. The sufficiency objective is usually pursued jointly with the RL objective.

While these methods can learn sufficient statistics and optimal policies in the context of POMDPs, they learn solely from the observations. However, assuming the same partial observability at training time and execution time is too pessimistic for many environments, notably for those that are simulated. We claim that additional information about the state s, be it partial or complete, can be leveraged during training for learning sufficient statistics more efficiently. To this end, we generalize the problem of learning from interaction in a POMDP by proposing the informed POMDP. This formalization introduces the training information i about the state s, which is only available at training time. Importantly, this training information is designed such that the observation is conditionally independent of the state given the information. Note that it is always possible to design such an information i, possibly by concatenating the observation o with the eventual additional observations o^+, such that $i = (o, o^+)$. This formalization offers a new learning paradigm where the training information is used along the reward and observation to supervise policy learning.

In this context, we prove that recurrent statistics are sufficient for optimal control when they are predictive sufficient for the reward and next information given the action: $p(r, i'|h, a) = p(r, i'|f(h), a)$. We then derive a learning objective for finding a predictive sufficient statistic, which amounts to approximating the conditional distribution $p(r, i'|h, a)$ through likelihood maximization using a model $q_\theta(r, i'|f_\theta(h), a)$, where f_θ is the recurrent statistic. Compared to the classic objective for learning sufficient statistics [Igl et al., 2018, Buesing et al., 2018, Han et al., 2019, Hafner et al., 2019], this objective approximates $p(r, i'|h, a)$ instead of $p(r, o'|h, a)$. Next, we show that this learned model $q_\theta(r, i'|f_\theta(h), a)$ can be adapted to provide a world model from which latent trajectories can be sampled without explicitly reconstructing the observation. This approach boils down to adapting latent world models such as those of PlaNet or Dreamer [Hafner et al., 2019, 2020, 2021, 2023] by relying on a model of the information instead of a model of the observation. Our claims are supported by experiments in several environments that we formalize as informed POMDPs (Mountain Hike, Velocity Control, Pop Gym, Flickering Atari and Flickering Control). The informed adaptation of Dreamer exhibits an improvement in terms of convergence speed and policy performance in many environments, while sometimes hurting performance in others.

This work is structured as follows. In Section 5.2, we present some related works in asymmetric learning and multi-agent RL. In Section 5.3, the informed POMDP is presented with the underlying execution POMDP. In Section 5.4, we provide a learning objective for sufficient statistics in this context. In Section 5.5, we adapt the Dreamer algorithm to informed POMDPs using this informed objective. In Section 5.6, we compare the Uninformed Dreamer and the Informed Dreamer in several environments.

5.2 Related Works

In RL for POMDPs, asymmetric learning consists of exploiting state information during training. These approaches usually learn policies for the POMDP by imitating a policy conditioned on the state [Choudhury et al., 2018]. However, these heuristic approaches lack a theoretical framework, and the resulting policies are known to be suboptimal for the POMDP [Warrington et al., 2021, Baisero et al., 2022]. Intuitively, optimal policies in POMDP might indeed need to consider actions that reduce state uncertainty. Warrington et al. [2021] addressed this issue by constraining the expert policy so that its imitation results in an optimal policy in the POMDP. Alternatively, asymmetric actor-critic approaches use a critic conditioned on the state [Pinto et al., 2018]. These approaches were proven to provide biased gradients by Baisero and Amato [2022], who also proposed an unbiased actor-critic approach by introducing the history-state value function $V(h, s)$. Baisero et al. [2022] adapted this method to value-based RL, where the history-dependent value function $V(h)$ uses the history-state value function $V(h, s)$ in its temporal difference target. Alternatively, Nguyen et al. [2021] proposed to enforce that the statistic $f(h)$ encodes the belief $p(s|h)$, a sufficient statistic for optimal control [Åström, 1965]. It makes the strong assumption that beliefs $b(s) = p(s|h)$ are available at training time. Finally, in a concurrent work, Avalos et al. [2024] learns a statistic $f(h)$ that encodes the

belief distribution $p(s|h)$ by leveraging the states during training.

In multi-agent RL, exploiting additional information available at training time was extensively studied under the centralized training and decentralized execution (CTDE) framework [Oliehoek et al., 2008]. In CTDE, it is assumed that the histories of all agents, or even the environment state, are available to all agents at training time. To exploit this additional information, several asymmetric actor-critic approaches have been developed by leveraging an asymmetric critic conditioned on all histories, including COMA [Foerster et al., 2018], MADDPG [Lowe et al., 2017], M3DDPG [Li et al., 2019] and R-MADDPG [Wang et al., 2020]. While efficient in practice, Lyu and Xiao [2022] showed that these asymmetric actor-critic approaches provide biased gradient estimates, which generalizes results developed for asymmetric learning in POMDP [Baisero and Amato, 2022] to the multi-agent setting. In the cooperative CTDE setting, another line of work focuses on value decomposition to learn a utility function for each agent, including QMIX [Rashid et al., 2018], QVMix [Leroy et al., 2021] and QPLEX [Wang et al., 2021]. These approaches use the additional information to modulate the contribution of each utility function in the global value function, while ensuring that maximizing the local utility functions also maximize the global value function, a property known as individual global max (IGM). Other methods relax this IGM requirement but still condition the value function on all histories, including QTRAN [Son et al., 2019] and WQMix [Rashid et al., 2020]. Recently, Hong et al. [2022] established that the IGM decomposition is not attainable in the general case.

In contrast to the existing literature on asymmetric learning in POMDP, we introduce an objective that provides a sufficient statistic for optimal control, and that leverages the additional information only through the objective. Moreover, our new learning paradigm is not restricted to state supervision, but supports any level of additional information. Finally, to the best of our knowledge, our method is the first to exploit additional information for learning an environment model of the POMDP. While our approach is probably applicable to the CTDE setting for learning sufficient statistics of the local histories of each agent, we leave it as future work.

5.3 Informed Partially Observable Markov Decision Processes

In this section, we introduce the informed POMDP and the associated training information, along with the underlying execution POMDP and the RL objective in this context.

5.3.1 Informed Partially Observable Markov Decision Processes

An informed POMDP $\widetilde{\mathcal{P}}$ is defined as a tuple $\widetilde{\mathcal{P}} = (\mathcal{S}, \mathcal{A}, \mathcal{I}, \mathcal{O}, T, R, \widetilde{I}, \widetilde{O}, P, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $\mathcal{I}$ is the information space, and $\mathcal{O}$ is the observation space. The initial state distribution P gives the probability $P(s_0)$ of $s_0 \in \mathcal{S}$ being the initial state of the decision process. The dynamics are described by the transition distribution T that gives the probability

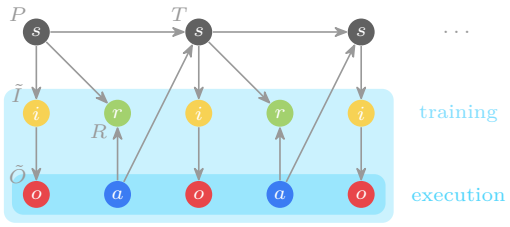


Figure 5.1: Bayesian network of an informed POMDP execution.

$T(s_{t+1}|s_t, a_t)$ of $s_{t+1} \in \mathcal{S}$ being the state resulting from action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$. The reward distribution R gives the probability density $R(r_t|s_t, a_t)$ of the immediate reward $r_t \in \mathbb{R}$ after taking action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$. The information distribution $\widetilde{I}$ gives the probability $\widetilde{I}(i_t|s_t)$ to get information $i_t \in \mathcal{I}$ in state $s_t \in \mathcal{S}$, and the observation distribution $\widetilde{O}$ gives the probability $\widetilde{O}(o_t|i_t)$ to get observation $o_t \in \mathcal{O}$ given information i_t. Finally, the discount factor $\gamma \in [0, 1]$ weights the relative importance of future rewards. The main assumption about an informed POMDP is that the observation o_t is conditionally independent of the state s_t given the information i_t: $p(o_t|i_t, s_t) = \widetilde{O}(o_t|i_t)$. In other words, the random variables s_t, i_t and o_t satisfy the Bayesian network $s_t \longrightarrow i_t \longrightarrow o_t$. In practice, it is always possible to define such a training information i_t. For example, the information $i_t = (o_t, o_t^+)$ satisfies the aforementioned conditional independence for any o_t^+. Taking a sequence of t actions in the informed POMDP conditions its execution and provides samples $(i_0, o_0, a_0, r_0, \ldots, i_t, o_t)$ at training time, as illustrated in Figure 5.1.

For each informed POMDP, there is an underlying execution POMDP that is defined as $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, P, \gamma)$, where $O(o_t|s_t) = \int_{\mathcal{I}} \widetilde{O}(o_t|i)\widetilde{I}(i|s_t)\,\mathrm{d}i$. Taking a sequence of t actions in the execution POMDP conditions its execution and provides the history $h_t = (o_0, a_0, \ldots, o_t) \in \mathcal{H}$, where $\mathcal{H}$ is the set of histories of arbitrary length. Note that information samples $i_0, \ldots, i_t$ and reward samples $r_0, \ldots, r_{t-1}$ are not included, since they are not available at execution time.

5.3.2 Reinforcement Learning Objective

A policy $\eta \in H$ is a mapping from histories to probability measures over the action space, where $H = \mathcal{H} \to \Delta(\mathcal{A})$ is the set of such mappings. A policy is said to be optimal for an informed POMDP when it is optimal in the underlying execution POMDP, i.e., when it maximizes the expected return,

$$J(\eta) = \mathbb{E}^{\eta}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right]. \tag{5.1}$$

The RL objective for an informed POMDP is thus to find an optimal policy $\eta^* \in \arg\max_{\eta \in H} J(\eta)$ for the execution POMDP from interaction with the informed POMDP.

5.4 Optimal Control with Recurrent Sufficient Statistics

In this section, we introduce the notion of sufficient statistic for optimal control and derive an objective for learning such a statistic in an informed POMDP. For the sake of conciseness, we simply use x to denote a random variable at the current time step and x' to denote it at the next time step. Moreover, we use the composition notation $g \circ f$ to denote the history-dependent policy $g(\cdot|f(\cdot))$.

5.4.1 Recurrent Sufficient Statistics

Let us first define the concept of sufficient statistic, and derive a necessary condition for optimality.

Definition 5.1 (Sufficient statistic). In an informed POMDP $\widetilde{\mathcal{P}}$ and in its underlying execution POMDP $\mathcal{P}$, a statistic of the history $f\colon \mathcal{H} \to \mathcal{Z}$ is sufficient for optimal control if, and only if,

$$\max_{g\colon \mathcal{Z}\to\Delta(\mathcal{A})} J(g \circ f) = \max_{\eta\colon \mathcal{H}\to\Delta(\mathcal{A})} J(\eta). \tag{5.2}$$

Corollary 5.1 (Sufficiency of optimal policies). In an informed POMDP $\widetilde{\mathcal{P}}$ and in its underlying execution POMDP $\mathcal{P}$, if a policy $\eta = g \circ f$ is optimal, then the statistic $f\colon \mathcal{H} \to \mathcal{Z}$ is sufficient for optimal control.

In this work, we focus on learning recurrent policies, i.e., policies $\eta = g \circ f$ for which the statistic f is recurrent. Formally, we have,

$$\eta(a|h) = g(a|f(h)),\ \forall (h, a), \tag{5.3}$$
$$f(h') = u(f(h), a, o'),\ \forall h' = (h, a, o'). \tag{5.4}$$

This enables the history to be processed iteratively each time that an action is taken and an observation is received. According to Corollary 5.1, when learning a recurrent policy $\eta = g \circ f$, the objective can be broken down into two problems: finding a sufficient statistic f and an optimal distribution g,

$$\max_{\substack{f\colon \mathcal{H}\to\mathcal{Z} \\ g\colon \mathcal{Z}\to\Delta(\mathcal{A})}} J(g \circ f). \tag{5.5}$$

5.4.2 Learning Recurrent Sufficient Statistics

Below, we provide a sufficient condition for a statistic to be sufficient for optimal control.

Theorem 5.1 (Sufficiency of recurrent predictive sufficient statistics). In an informed POMDP $\widetilde{\mathcal{P}}$, a statistic $f\colon \mathcal{H} \to \mathcal{Z}$ is sufficient for optimal control if it is (i) recurrent and (ii) predictive sufficient for the reward and next information given the action,

$$\text{(i) } f(h') = u(f(h), a, o'),\ \forall h' = (h, a, o'), \tag{5.6}$$
$$\text{(ii) } p(r, i'|h, a) = p(r, i'|f(h), a),\ \forall (h, a, r, i'). \tag{5.7}$$

The proof for this theorem is in Appendix 5.A, generalizing earlier work by Subramanian et al. [2022].

Now, let us consider a distribution over the histories and actions whose density function is denoted as $p(h, a)$. For example, we consider the stationary distribution induced by the current policy η in the informed POMDP $\widetilde{\mathcal{P}}$. Let us also assume that the density function $p(h, a)$ is non-zero everywhere. As shown in Appendix 5.B, under mild assumptions, any statistic f satisfying the objective,

$$\max_{\substack{f\colon \mathcal{H}\to\mathcal{Z} \\ q\colon \mathcal{Z}\times\mathcal{A}\to\Delta(\mathbb{R}\times\mathcal{I})}} \mathop{\mathbb{E}}_{p(h,a,r,i')} \log q(r, i'|f(h), a) \tag{5.8}$$

also satisfies (ii). This variational objective jointly optimizes the statistic function $f\colon \mathcal{H} \to \mathcal{Z}$ with a conditional probability density function $q\colon \mathcal{Z} \times \mathcal{A} \to \Delta(\mathbb{R} \times \mathcal{I})$. According to Theorem 5.1, a statistic that is recurrent and that satisfies objective (5.8) is sufficient for optimal control.

In practice, both the recurrent statistic and the density function are implemented with neural networks f_θ and q_θ respectively, both parametrized by $\theta \in \mathbb{R}^d$. In this case, the objective can be maximized by stochastic gradient ascent. Regarding the statistic function f_θ, it is implicitly implemented by the update function $z_t = u_\theta(z_{t-1}; x_t)$ of an RNN. The inputs are $x_t = (a_{t-1}, o_t)$, with a_{-1} the null action that is typically set to zero. The hidden state of the RNN $z_t = f_\theta(h_t)$ is thus a statistic of the history that is recurrently updated using u_θ. Regarding q_θ, it is implemented by a parametrized probability density function estimator. In such a context, we obtain the objective,

$$\max_\theta \underbrace{\mathop{\mathbb{E}}_{p(h,a,r,i')} \log q_\theta(r, i'|f_\theta(h), a)}_{L(f_\theta)}. \tag{5.9}$$

We might wonder whether this informed objective is better than the classic objective, where $i = o$. In this work, we hypothesize that approximating the information distribution instead of the observation distribution is a better objective in practice. This is motivated by the data processing inequality applied to the Bayesian network $s' \longrightarrow i' \longrightarrow o'$, which concludes that the information i' is more informative than the observation o' about the Markovian state s' of the environment,

$$I(s', i'|h, a) \geq I(s', o'|h, a), \tag{5.10}$$

where I denotes the conditional mutual information. We thus expect the statistic $f_\theta(h)$ to converge faster towards a sufficient statistic, and the policy to converge faster towards an optimal policy. It is however important to note that the information i might contain irrelevant state variables. In practice, the conditional distribution $p(i'|h, a)$ may thus be much more difficult to approximate than $p(o'|h, a)$, while not being much more useful to the control task. While we consider this study out of the scope of this work, ensuring that the sufficient representations of the histories are also necessary for the control task is a promising avenue for future work.

5.4.3 Optimal Control with Recurrent Sufficient Statistics

As seen from Corollary 5.1, sufficient statistics are needed for optimally controlling POMDPs. Moreover, as we focus on recurrent policies implemented with RNNs, we can exploit objective (5.9) to learn a sufficient statistic f_θ. In practice, we jointly maximize the RL objective $J(\eta_{\theta,\phi}) = J(g_\phi \circ f_\theta)$ and the statistic objective $L(f_\theta)$. This enables one to use the information i to guide the statistic learning through $L(f_\theta)$. This joint maximization results in the objective,

$$\max_{\theta,\phi} J(g_\phi \circ f_\theta) + L(f_\theta). \tag{5.11}$$

Note that a policy maximizing (5.11) also maximizes the return $J(g_\phi \circ f_\theta)$ if f_θ and q_θ are expressive enough, such that this objective provides optimal policies in the sense of objective (5.5).

5.5 Model-Based RL with Informed World Models

Model-based RL focuses on learning a model of the dynamics $p(r, o'|h, a)$ of the environment, known as a world model, that is exploited to derive a near-optimal policy. Since the approximate model usually allows one to generate trajectories, many works derive a near-optimal policy by online planning (e.g., model-predictive control) or by optimizing a parametrized policy based on these trajectories [Sutton, 1991, Ha and Schmidhuber, 2018, Chua et al., 2018, Zhang et al., 2019, Hafner et al., 2019, 2020]. In this section, we first modify the model $q_\theta(r, i'|f_\theta(h), a)$ in order to get a world model from which trajectories can be sampled. We then adapt the DreamerV3 [Hafner et al., 2023] algorithm using this world model, resulting in the Informed Dreamer algorithm.

5.5.1 Informed World Model

We implement the informed world model with a variational RNN (VRNN) as introduced by Chung et al. [2015], also known as a recurrent state-space model (RSSM) in the RL context [Hafner et al., 2019]. It is worth noticing that such a model performs its recurrent update using a latent stochastic representation of the observation. When generating trajectories, it also samples latent representations of the observations without explicitly reconstructing them, which we refer to as latent trajectories. This key design choice enables the sampling of trajectories without explicitly learning the observation distribution, but the reward and information distribution only. Formally, we have,

$$\hat{e} \sim q^p_\theta(\cdot|z, a), \qquad \text{(prior, 5.12)}$$
$$\hat{r} \sim q^r_\theta(\cdot|z, \hat{e}), \qquad \text{(reward decoder, 5.13)}$$
$$\hat{i}' \sim q^i_\theta(\cdot|z, \hat{e}), \qquad \text{(information decoder, 5.14)}$$

where $\hat{e}$ is the latent variable of the VRNN when generating trajectories. The prior q^p_θ and the decoders q^i_θ and q^r_θ are jointly trained with the encoder,

$$e \sim q^e_\theta(\cdot|z, a, o'), \qquad \text{(encoder, 5.15)}$$

to maximize the likelihood of reward and next information samples. The latent representation $e \sim q^e_\theta(\cdot|z, a, o')$ of the next observation o' can be used to update the statistic to z',

$$z' = u_\theta(z, a, e). \qquad \text{(recurrence, 5.16)}$$

Note that the statistic z is no longer deterministically updated to z' given a and o', instead we have $z \sim f_\theta(\cdot|h)$, which is induced by u_θ and q^e_θ. In practice, we maximize the evidence lower bound (ELBO), a variational lower bound on the likelihood of reward and next information samples given the statistic [Chung et al., 2015],

$$\mathop{\mathbb{E}}_{\substack{p(h,a,r,i') \\ f_\theta(z|h)}} \log q_\theta(r, i'|z, a) \geq \mathop{\mathbb{E}}_{\substack{p(h,a,r,i',o') \\ f_\theta(z|h)}} \Bigg[\mathop{\mathbb{E}}_{q^e_\theta(e|z,a,o')} \left[\log q^i_\theta(i'|z, e) + \log q^r_\theta(r|z, e) \right] - \mathrm{KL}\left(q^e_\theta(\cdot|z, a, o') \parallel q^p_\theta(\cdot|z, a) \right) \Bigg]. \quad (5.17)$$

As illustrated in Figure 5.2 for a trajectory sampled in the informed POMDP, the ELBO objective maximizes the conditional log-likelihood $q^r_\theta(r|z, e)$ and $q^i_\theta(i|z, e)$ of r and i' for a sample of the encoder $e \sim q^e_\theta(\cdot|z, a, o')$, and minimizes the KL divergence from $q^e_\theta(\cdot|z, a, o')$ to the prior distribution $q^p_\theta(\cdot|z, a)$. Note that when $i = o$, it corresponds to Dreamer's world model and learning objective.

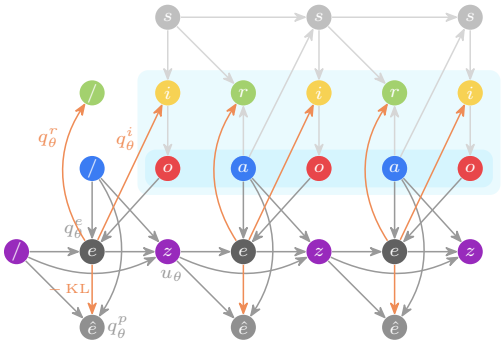


Figure 5.2: Variational recurrent neural network loss for a given trajectory at training time. Dependence of q^r_θ and q^i_θ on z is omitted.

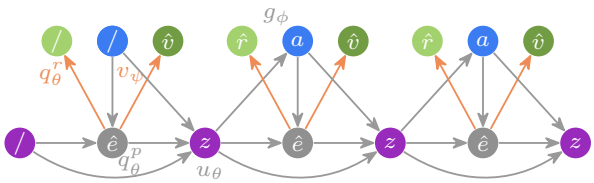
As can be noticed from Equation 5.17 and Figure 5.2, the encoder is conditioned on the observation and not on the information. While this is required for the encoder to be used at execution time, it certainly loosen the lower bound and limits the quality of the conditional information distribution that can be learned. Future work may improve the quality of the information reconstruction by considering an additional information encoder, also conditioned on the statistic of the history, whose samples are not used in the recurrence.

5.5.2 Informed Dreamer

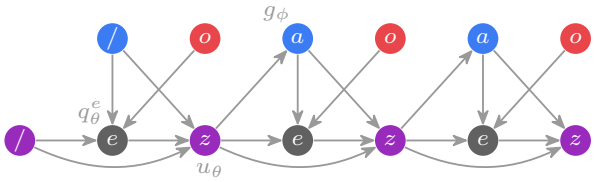
As explained above, while our informed world model does not learn the observation distribution, it is still able to sample latent trajectories. Indeed, the VRNN only uses the latent representation $e \sim q^e_\theta(\cdot|z, a, o')$ of the observation o', trained

to reconstruct the information i', in order to update z to z'. Consequently, we can use the prior distribution $\hat{e} \sim q^p_\theta(\cdot|z, a)$, trained according to (5.17) to minimize the KL divergence from $e \sim q^p_\theta(\cdot|z, a, o')$ in expectation, to sample latent trajectories.

The Informed Dreamer algorithm leverages such trajectories to learn a latent critic $v_\psi(z)$ and a latent policy $a \sim g_\phi(\cdot|z)$. Figure 5.3a illustrates the generation of a latent trajectory, along with estimated rewards $\hat{r} \sim q^r_\theta(\cdot|z, e)$ and values $\hat{v} = v_\psi(z)$. The actions are sampled according to the latent policy, and any RL algorithm can be used to maximize the estimated return. Moreover, note that the estimated return is given by a function that is differentiable with respect to ϕ, and it can be directly maximized by stochastic gradient ascent. In the experiments, we use an actor-critic approach for discrete actions and direct maximization for continuous actions, following DreamerV3 [Hafner et al., 2023]. Finally, as shown in Figure 5.3b, when deployed in the execution POMDP, the encoder q^e_θ is used to compute the latent representations of the observations and to update the statistic. The actions are then selected according to $a \sim g_\phi(\cdot|z)$.



(a) Imagination of a trajectory using policy g_ϕ with estimated rewards and values. Dependence of q^r_θ and v_ψ on z is omitted.



(b) Execution of the policy on a trajectory of the POMDP using the encoder q^e_θ to condition the latent policy g_ϕ.

Figure 5.3: Bayesian graph of a VRNN evaluation during imagination and execution.

A pseudocode for the adaptation of the DreamerV3 algorithm using this informed world model is given in Appendix 5.C. We also detail some divergences of our formalization with respect to the original DreamerV3 algorithm. As in DreamerV3, we use symlog predictions, a discrete VAE, KL balancing, free bits, reward normalization, a distributional critic, and entropy regularization.

5.6 Experiments

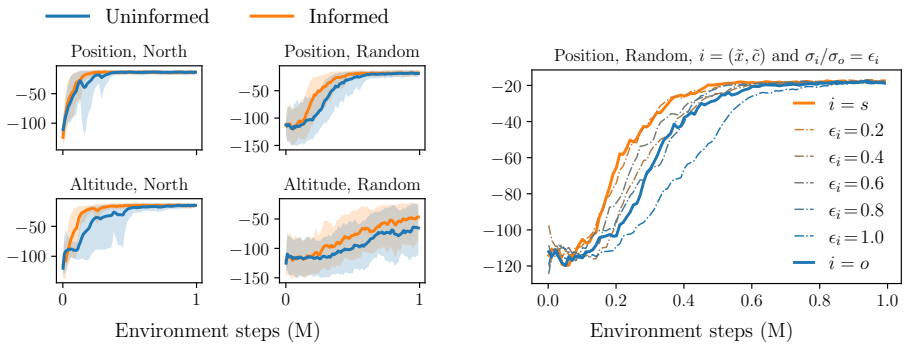
In this section, we compare Dreamer to the Informed Dreamer on several informed POMDPs, all considered with a discount factor of $\gamma = 0.997$. For reproducibility purposes, we use the implementation and hyperparameters of

DreamerV3 released by the authors at github.com/danijar/dreamerv3, and release our adaptation to informed POMDPs using the same hyperparameters at github.com/glambrechts/informed-dreamer.

5.6.1 Varying Mountain Hike

In the Varying Mountain Hike environments, the agent should walk throughout a mountainous terrain to reach the mountain top as fast as possible while avoiding the valleys. There exists four versions of this environment, depending on the agent orientation (*north* or *random*) and on the observation that is available (*position* or *altitude*). More formally, the agent has a position x and a fixed orientation c in each episode. The orientation c is either always *north* or a *random* cardinal orientation, depending on the environment version. It can take four actions to move relative to its orientation (right, forward, left and backward). The orientation is not observed by the agent, but it receives a Gaussian observation of its *position*, or its *altitude*, depending on the environment version ($\sigma_o = 0.1$ in both cases). The reward is given by its altitude relative to the mountain top, such that the goal of the agent is to obtain the highest cumulative altitude. Around the mountain top, states are terminal and the trajectories are truncated at $t = 160$ in practice. We refer the reader to Lambrechts et al. [2022] for a formal description of these environments, strongly inspired by the Mountain Hike of Igl et al. [2018].

For this environment, we first consider the position and orientation to be available as additional information at training time. In other words, we consider the state-informed POMDP with $i = s$. As can be seen in Figure 5.4a, the speed of convergence of the policies is improved in all four environments when using the Informed Dreamer. Moreover, as shown in Table 5.1 in Appendix 5.D, the final performance of the Informed Dreamer is better in 3 out of 4 environments.



(a) Uninformed Dreamer and Informed Dreamer with $i = s$ in the four environments.

(b) Informed Dreamer with $i = (\tilde{x}, \tilde{c})$ with position observation and random orientation.

Figure 5.4: Varying Mountain Hike environments: minimum, maximum and average returns over five trainings.

We also experiment with other types of information in the Varying Mountain Hike with position observation and random orientation. More precisely, we consider an information $i = (\tilde{x}, \tilde{c})$ about the state $s = (x, c)$, where $\tilde{x}$ is an

observation of the position x with Gaussian noise of standard deviation $\sigma_i \in [0, \sigma_o]$, and $\tilde{c}$ is a noisy observation of the orientation c replaced by a random orientation with probability $\varepsilon_i \in [0, 1]$. Note that when $\sigma_i = 0$, the position x is encoded in the information, while when $\sigma_i = \sigma_o$, the observation o is encoded in the information. As shown in Figure 5.4b, without confidence intervals for the sake readability, the better the information, the faster the policy converges. It supports the idea that the more information about the state is exploited, the faster an optimal policy for the POMDP is learned. Moreover, we observe that the Informed Dreamer with $\varepsilon_i = 1$ and $\sigma_i = 0.1$ performs even worse than the Uninformed Dreamer. It suggests that considering additional information that is not informative about the state (i.e., $I(s, i|o) = 0$), such as $\tilde{c}$ with $\varepsilon_i = 1$, can degrade learning. Similar results are obtained for the other three environments in Subappendix 5.E.1.

5.6.2 Velocity Control

In the Velocity Control environments, we consider the standard DeepMind Control tasks [Tassa et al., 2018], where only the joints velocities are available as observations and not their absolute positions, which is a standard benchmark in the partially observable RL literature [Han et al., 2019, Lee et al., 2020]. These environments consists of controlling different multi-joints robots to achieve several tasks. We consider the absolute positions to be available at training time along with the velocities, which results in a Markovian information $i = s$.

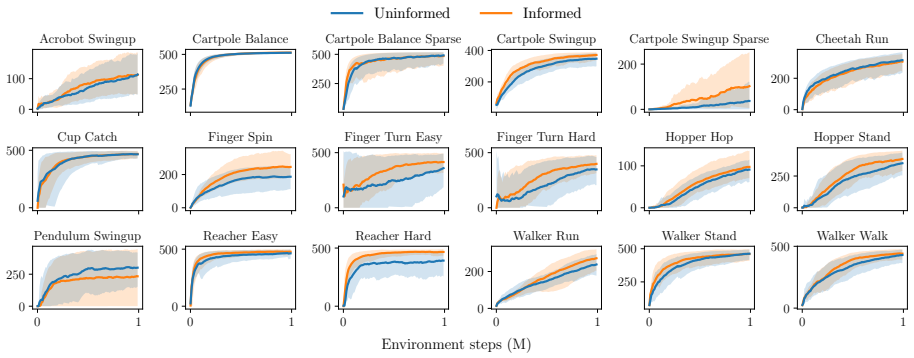


Figure 5.5: Uninformed Dreamer and Informed Dreamer with $i = s$ in the Velocity Control environments: minimum, maximum and average returns over five trainings.

Figure 5.5 shows that the convergence speed of the policies is improved in this benchmark, for nearly all of the considered games. Moreover, the final returns are given in Table 5.2 in Appendix 5.D, and show that policies obtained after one million time steps are better in 13 out of 18 environments when considering additional information.

5.6.3 Pop Gym

The Pop Gym environments have been specifically designed to benchmark the ability of handling partial observability [Morad et al., 2023]. The latter notably

includes memory games, board games, or control problems involving partial observability and noise. For these environments, we consider the state to be available as additional information.

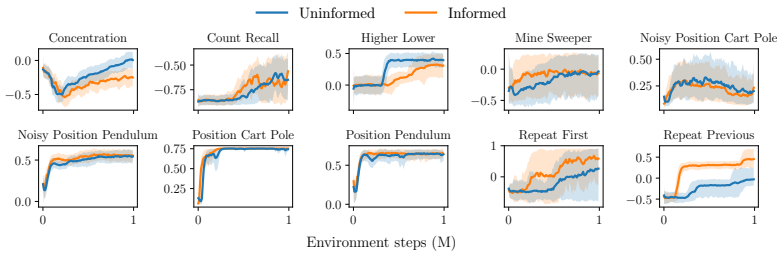


Figure 5.6: Uninformed Dreamer and Informed Dreamer with $i = s$ in the Pop Gym environments: minimum, maximum and average returns over five trainings.

Figure 5.6 shows that learning in those POMDPs sometimes benefits from the exploitation of additional information as proposed in the Informed Dreamer. The learning of the Informed Dreamer seems to suffer from the approximation of the information distribution in 2 out of those 10 environments (Concentration and Higher Lower). The final returns are given in Table 5.3 in Appendix 5.D, showing a better final performance in 7 out of 10 environments, even though returns have a high variability. In particular, we observe that the Informed Dreamer converges to a significantly higher return for the Repeat First and Repeat Previous environments, that both require discovering long time dependencies. The exploitation of additional information seems crucial in these environments, and we study this in depth on harder instances of the Repeat Previous environment in Subappendix 5.E.2. This analysis shows that the Informed Dreamer can learn near-optimal policies in environments for which the Uninformed Dreamer does not learn at all.

5.6.4 Flickering Atari and Flickering Control

While arguably not constituting a relevant benchmark for measuring the ability of handling partial observability [Shao et al., 2022, Avalos et al., 2024], the Flickering Atari and Flickering Control environments have become standard benchmarks in the partially observable RL literature [Hausknecht and Stone, 2015, Zhu et al., 2017, Igl et al., 2018, Ma et al., 2020]. For completeness, the results for these environments are reported in Appendix 5.E. We observe that the speed of convergence and final performance of the agent is sometimes greatly improved when considering additional information (e.g., Asteroids, Pong, Breakout). However, we also observe that the performance is lower in some environments. As far as the Flickering Atari environments are concerned, the Informed Dreamer only outperforms Dreamer in 6 out of 12 environments. In the Flickering Control environments, the Informed Dreamer tends to systematically underperform the Uninformed Dreamer, attaining a better performance in only 2 out of 18 environments. It suggests that additional state information is not useful for these tasks. We furthermore hypothesize that the conditional information distribution is difficult to approximate, which may cause learning to degrade. It

shows that not all information is worth exploiting, particularly when the level of uncertainty due to partial observability is low.

5.7 Conclusion

In this work, we introduced a new formalization for considering additional information available at training time for POMDP, called the informed POMDP. In this context, we proposed a learning objective and proved that it provides sufficient statistic for optimal control. Next, we adapted this objective to provide an environment model from which latent trajectories can be sampled. We then adapted a successful model-based RL algorithm, known as Dreamer, with this informed world model, resulting in the Informed Dreamer algorithm. By considering several environments from the partially observable RL literature, we showed that this informed learning objective often improves the convergence speed and quality of the policies. This work also presents several limitations. First, a formal justification for the use of the information instead of the observation is still lacking. Future work may consider the notion of approximate information states to bound the suboptimality of the policy for a given error on the information distribution instead of the observation distribution. Second, we observed that this informed objective hurts performance in some environments, motivating further work in which particular attention is paid to the design of the information. It would be worth drawing connection to the exogenous RL literature that complements this work by focusing on discarding irrelevant information. Third, the proposed ELBO learning objective is probably a loose lower bound on the information likelihood. Future work may improve the quality of the information distribution by considering informed world models with a dedicated information encoder.

5.A Proof of the Sufficiency of Recurrent Predictive Sufficient Statistics

In this section, we prove Theorem 5.1, that is recalled below.

Theorem 5.1 (Sufficiency of recurrent predictive sufficient statistics). In an informed POMDP $\widetilde{\mathcal{P}}$, a statistic $f\colon \mathcal{H} \to \mathcal{Z}$ is sufficient for optimal control if it is (i) recurrent and (ii) predictive sufficient for the reward and next information given the action,

$$\text{(i) } f(h') = u(f(h), a, o'),\ \forall h' = (h, a, o'), \tag{5.6}$$
$$\text{(ii) } p(r, i'|h, a) = p(r, i'|f(h), a),\ \forall (h, a, r, i'). \tag{5.7}$$

Proof. From Proposition 4 and Theorem 5 by Subramanian et al. [2022], we know that a statistic is sufficient for optimally controlling an execution POMDP if it is (i) recurrent and (ii') predictive sufficient for the reward and next *observation* given the action: $p(r, o'|h, a) = p(r, o'|f(h), a)$. Let us consider a statistic $f\colon \mathcal{H} \to \mathcal{A}$ satisfying (i) and (ii). Let us show that it satisfies (ii'). We have,

$$p(r, o'|f(h), a) = \int_{\mathcal{I}} p(r, o', i'|f(h), a)\,\mathrm{d}i' \tag{5.18}$$
$$= \int_{\mathcal{I}} p(o'|r, i', f(h), a) p(r, i'|f(h), a)\,\mathrm{d}i', \tag{5.19}$$

using the law of total probability and the chain rule. As can be seen from the informed POMDP formalization of Section 5.3 and the resulting Bayesian network in Figure 5.1, the Markov blanket of o' is $\{i'\}$. As a consequence, o' is conditionally independent of any other variable given i'. In particular, $p(o'|i', r, f(h), a) = p(o|i')$, such that,

$$p(r, o'|f(h), a) = \int_{\mathcal{I}} p(o'|i') p(r, i'|f(h), a)\,\mathrm{d}i'. \tag{5.20}$$

From hypothesis (ii), we can write,

$$p(r, o'|f(h), a) = \int_{\mathcal{I}} p(o'|i') p(r, i'|h, a)\,\mathrm{d}i'. \tag{5.21}$$

Finally, exploiting the Markov blanket $\{i'\}$ of o', the chain rule and the law of total probability again, we have,

$$p(r, o'|f(h), a) = \int_{\mathcal{I}} p(o'|i', r, h, a) p(r, i'|h, a)\,\mathrm{d}i' \tag{5.22}$$
$$= \int_{\mathcal{I}} p(o', r, i'|h, a)\,\mathrm{d}i' \tag{5.23}$$
$$= p(r, o'|h, a). \tag{5.24}$$

This proves that (ii) implies (ii'). As a consequence, any statistic f satisfying (i) and (ii) is a sufficient statistic of the history for optimally controlling the informed POMDP. □

5.B Proof of the Predictive Sufficient Objective

First, let us consider a fixed history h and action a. Let us recall that two density functions $p(r, i'|h, a)$ and $p(r, i'|f(h), a)$ are equal almost everywhere if, and only if, their KL divergence is zero,

$$\mathop{\mathbb{E}}_{p(r,i'|h,a)} \log \frac{p(r, i'|h, a)}{p(r, i'|f(h), a)} = 0. \tag{5.25}$$

Now, let us consider a probability density function $p(h, a)$ that is non zero everywhere. We have that the KL divergence from $p(r, i'|h, a)$ to $p(r, i'|f(h), a)$ is equal to zero for almost every history h and action a if, and only if, it is zero on expectation over $p(h, a)$ since the KL divergence is non-negative,

$$\mathop{\mathbb{E}}_{p(r,i'|h,a)} \log \frac{p(r, i'|h, a)}{p(r, i'|f(h), a)} \overset{\text{a.e.}}{=} 0 \Leftrightarrow \mathop{\mathbb{E}}_{p(h,a,r,i')} \log \frac{p(r, i'|h, a)}{p(r, i'|f(h), a)} = 0. \tag{5.26}$$

Rearranging, we have that $p(r, i'|h, a)$ is equal to $p(r, i'|f(h), a)$ for almost every h, a, r and i' if, and only if,

$$\mathop{\mathbb{E}}_{p(h,a,r,i')} \log p(r, i'|h, a) = \mathop{\mathbb{E}}_{p(h,a,r,i')} \log p(r, i'|f(h), a). \tag{5.27}$$

Now, we recall the data processing inequality, enabling one to write, for any statistic f',

$$\mathop{\mathbb{E}}_{p(h,a,r,i')} \log p(r, i'|h, a) \geq \mathop{\mathbb{E}}_{p(h,a,r,i')} \log p(r, i'|f'(h), a). \tag{5.28}$$

since $h(r, i'|h, a) = h(r, i'|h, f(h), a) \leq h(r, i'|f(h), a),\ \forall (h, a)$, where $h(x)$ is the differential entropy of random variable x. Assuming that there exists at least one $f\colon \mathcal{H} \to \mathcal{Z}$ for which the inequality is tight, we obtain the following objective for a predictive sufficient statistic f,

$$\max_{f\colon \mathcal{H}\to\mathcal{Z}} \mathop{\mathbb{E}}_{p(h,a,r,i')} \log p(r, i'|f(h), a). \tag{5.29}$$

Unfortunately, the probability density $p(r, i'|f(h), a)$ is unknown. However, knowing that the distribution that maximizes the log-likelihood of samples from $p(r, i'|f(h), a)$ is $p(r, i'|f(h), a)$ itself, we can write,

$$\mathop{\mathbb{E}}_{p(h,a,r,i')} \log p(r, i'|f(h), a) = \max_{q\colon \mathcal{Z}\times\mathcal{A}\to\Delta(\mathbb{R}\times\mathcal{I})} \mathop{\mathbb{E}}_{p(h,a,r,i')} \log q(r, i'|f(h), a). \tag{5.30}$$

By jointly maximizing the probability density function $q\colon \mathcal{Z} \times \mathcal{A} \to \Delta(\mathbb{R} \times \mathcal{I})$, we obtain,

$$\max_{\substack{f\colon \mathcal{H}\to\mathcal{Z} \\ q\colon \mathcal{Z}\times\mathcal{A}\to\Delta(\mathbb{R}\times\mathcal{I})}} \mathop{\mathbb{E}}_{p(h,a,r,i')} \log q(r, i'|f(h), a). \tag{5.31}$$

This objective ensures that the statistic $f(h)$ is predictive sufficient for the reward and next information given the action. If $f(h)$ is a recurrent statistic, then it is also sufficient for optimal control, according to Theorem 5.1.

5.C Informed Dreamer

The Informed Dreamer algorithm is presented in Algorithm 5.1. Differences with the Uninformed Dreamer algorithm [Hafner et al., 2020] are highlighted in blue. In addition, it can be noted that in the original Dreamer algorithm, the statistic z_t encodes $h_t = (o_0, a_0, \dots, o_t)$ and a_t, instead of h_t only. As a consequence, the prior distribution $e_t \sim q^p_\theta(\cdot|z_t)$ can be conditioned on the statistic z_t only, instead of the statistic and last action. Similarly, the encoder distribution $e_t \sim q^p_\theta(\cdot|z_t, o_{t+1})$ can be conditioned on the statistic z_t only, instead of the statistic and last action. On the other hand, the latent policy $a_{t+1} \sim g(\cdot|z_t, e_t)$ should be conditioned on the statistic z_t and the new latent e_t to account for the last observation, and the same is true for the value function $v_\psi(z_t, e_t)$. In the experiments, we follow the original implementation for both the Uninformed Dreamer and the Informed Dreamer, according to the code that we release at github.com/glambrechts/informed-dreamer.

Following Dreamer, the algorithm introduces the continuation flag c_t, which indicates whether state s_t is terminal. A terminal state s_t is a state from which the agent can never escape, and in which any further action provides a zero reward. It follows that the value function of a terminal state is zero, and trajectories can be truncated at terminal states since we do not need to learn their value or the optimal policy in those states. Alternatively, c_t can be interpreted as an indicator that can be extracted from the observation o_t, but we made it explicit in the algorithm.

5.D Final Returns

We provide the final returns obtained by Dreamer and the Informed Dreamer for the Varying Mountain Hike environments in Table 5.1, for the Velocity Control environments in Table 5.2, and for the Pop Gym environments in Table 5.3.

5.E Additional Experiments

In this section, we provide results for non-Markovian information in the Varying Mountain Hike environments, for harder Pop Gym environments, along with the results of the flickering environments.

5.E.1 Non-Markovian Information

We experiment with other levels of information in the Varying Mountain Hike environments. More precisely, we consider an information i that contains an observation $\tilde{x}$ of the position x (or an observation $\tilde{y}$ of the altitude y) with Gaussian noise of standard deviation $\sigma_i \in [0, \sigma_o]$. In addition, in the case of environments with random orientation, we consider an information that also contains a noisy observation of the orientation c replaced with a random orientation with probability $\varepsilon_i \in [0, 1]$. Note that when $\sigma_i = 0$, the exact position x (or altitude y) is encoded in the information, while when $\sigma_i = \sigma_o$, the observation o is encoded in the information.

As shown in Figure 5.7, without confidence intervals for the sake of readability,

Algorithm 5.1: Informed Dreamer.

parameters: S the number of environment steps, F the number of steps before training,
R the train ratio,
W the backpropagation horizon,
K the imagination horizon,
N the batch size,
B replay buffer capacity.

Initialize parameters θ, ϕ, ψ randomly, and empty replay buffer $\mathcal{B}$.
Let $g = 0$, $t = 0$, $a_{-1} = 0$, $r_{-1} = 0$, $z_{-1} = 0$.
Reset the environment and observe o_0 and c_0 (true at reset).
for $s = 0 \dots S-1$ **do**
  Encode observation o_t to $e_{t-1} \sim q^e_\theta(\cdot|z_{t-1}, a_{t-1}, o_t)$.
  Update $z_t = u_\theta(z_{t-1}, a_{t-1}, e_{t-1})$.
  Given the current statistic z_t, take action $a_t \sim g_\phi(\cdot|z_t)$.
  Observe reward r_t, information i_{t+1}, observation o_{t+1} and continuation flag c_{t+1}.
  if c_{t+1} *is false (terminal state)* **then**
    Reset $t = 0$.
    Reset the environment and observe o_0 and c_0 (true at reset).
  Update $t = t + 1$.
  Add trajectory of W steps $(a_{w-1}, r_{w-1}, i_w, o_w, c_w)_{w=t-W+1}^{t}$ to buffer $\mathcal{B}$.
  while $|\mathcal{B}| \geq F \wedge g < Rs$ **do**
    Draw N trajectories of length W $\left\{(a^n_{w-1}, r^n_{w-1}, i^n_w, o^n_w, c^n_w)_{w=0}^{W-1}\right\}_{n=0}^{N-1}$ uniformly from replay buffer $\mathcal{B}$.
    Compute statistics and encoded latents

$$\left\{(z^n_w, e^n_w)_{w=-1}^{W-2}\right\}_{n=0}^{N-1} = \text{Encode}\left(u_\theta, q^e_\theta, \left\{(a^n_{w-1}, o^n_w)_{w=0}^{W-1}\right\}_{n=0}^{N-1}\right).$$

    Update θ using $\nabla_\theta \sum_{n=0}^{N} \sum_{w=-1}^{W-2} L^n_w$, where $a^n_{-1} = 0$ and,

$$L^n_w = \log q^i_\theta(i^n_{w+1}|z^n_w, e^n_w) + \log q^c_\theta(c^n_{w+1}|z^n_w, e^n_w) + \log q^r_\theta(r^n_w|z^n_w, e^n_w) - \mathrm{KL}\left(q^e_\theta(\cdot|z^n_w, a^n_w, o^n_{w+1}) \parallel q^p_\theta(\cdot|z^n_w, a^n_w)\right).$$

    Sample latent trajectories

$$\left\{\left\{(z^{n,w}_k, \hat{e}^{n,w}_k)_{k=0}^{K-1}\right\}_{w=-1}^{W-2}\right\}_{n=0}^{N-1} = \text{Imagine}\left(u_\theta, q^p_\theta, g_\phi, \left\{(z^n_w, e^n_w, a^n_w)_{w=-1}^{W-2}\right\}_{n=0}^{N-1}\right).$$

    Predict rewards $r^{n,w}_k \sim q^r_\theta(\cdot|z^{n,w}_k, \hat{e}^{n,w}_k)$, continuations flags $c^{n,w}_{k+1} \sim q^c_\theta(\cdot|z^{n,w}_k, \hat{e}^{n,w}_k)$, and values $v^{n,w}_k = v_\psi(z^{n,w}_k)$.
    Compute value targets using λ-returns, with $G^{n,w}_{K-1} = v^{n,w}_{K-1}$ and

$$G^{n,w}_k = r^{n,w}_k + \gamma c^{n,w}_k \left((1-\lambda) v^{n,w}_{k+1} + \lambda G^{n,w}_{k+1}\right).$$

    Update ϕ using $\nabla_\phi \sum_{n=0}^{N-1} \sum_{w=-1}^{W-2} \sum_{k=0}^{K-1} G^{n,w}_k$.
    Update ψ using $\nabla_\psi \sum_{n=0}^{N-1} \sum_{w=-1}^{W-2} \sum_{k=0}^{K-1} \|v_\psi(z^{n,w}_k) - \mathrm{sg}(G^{n,w}_k)\|^2$, where sg is the stop-gradient operator.
    Count gradient steps $g = g + 1$.

Algorithm 5.2: Trajectory encoding.

inputs: u_θ the update function,
q^e_θ the encoder,
$\left\{(a^n_{w-1}, o^n_w)^{W-1}_{w=0}\right\}^{N-1}_{n=0}$ the histories.

Let $z^n_{-1} = 0$.
for $w = 0 \dots W-1$ **do**
  Let $e^n_{w-1} \sim q^e_\theta(\cdot|z^n_{w-1}, a^n_{w-1}, o^n_w)$.
  Let $z^n_w = u_\theta(z^n_{w-1}, a^n_{w-1}, e^n_{w-1})$.
return $\left\{(z^n_w, e^n_w)^{W-2}_{w=-1}\right\}^{N-1}_{n=0}$.

Algorithm 5.3: Trajectory imagination.

inputs: u_θ the update function,
q^p_θ the prior,
g_ϕ the policy,
$\left\{(z^n_w, e^n_w, a^n_w)^{W-2}_{w=-1}\right\}^{N-1}_{n=0}$ the statistics, encoded latents and actions.

Let $z^{n,w}_{-1} = z^n_w$, $\hat{e}^{n,w}_{-1} = e^n_w$, $a^{n,w}_{-1} = a^n_w$.
for $k = 0 \dots K-1$ **do**
  Let $z^{n,w}_k = u_\theta(z^{n,w}_{k-1}, a^{n,w}_{k-1}, \hat{e}^{n,w}_{k-1})$.
  Let $\hat{e}^{n,w}_k \sim q^p_\theta(\cdot|z^{n,w}_k, a^{n,w}_k)$.
  Let $a^{n,w}_k \sim g_\phi(\cdot|z^{n,w}_k)$.
return $\left\{\left\{(z^{n,w}_k, \hat{e}^{n,w}_k)^{K-1}_{k=0}\right\}^{W-2}_{w=-1}\right\}^{N-1}_{n=0}$.

Altitude	Random	Uninformed	Informed
False	False	-13.70 ± 03.32	$\mathbf{-13.35 \pm 02.93}$
False	True	-18.32 ± 06.04	$\mathbf{-17.72 \pm 04.19}$
True	False	$\mathbf{-14.78 \pm 02.44}$	-14.98 ± 04.73
True	True	-67.05 ± 21.76	$\mathbf{-45.94 \pm 32.77}$

Table 5.1: Average final return and standard deviation over five trainings in the Mountain Hike environments.

the better the information, the faster the policy converges. These results hold in all environments except that with altitude observation and fixed orientation, for which the results are more mixed. As said in Subsection 5.6.1, it supports the hypothesis that the more informative about the state the information is, the faster an optimal policy is learned. Moreover, it can be observed that when an additional information $\tilde{c}$ is not informative about the state, convergence is slower than for the Uninformed Dreamer. This highlights again the importance of the quality of the additional information.

5.E.2 Harder Pop Gym Environments

Despite the performance of the informed policy being equal to the performance of the uninformed policy at optimum, there may exists environments for which the optimum is never reached in practice without considering additional information at training time. We observe it to be the case for environments with long

Task	Uninformed	Informed
Acrobot Swingup	$\mathbf{113.73 \pm 108.03}$	112.49 ± 54.67
Cartpole Balance	511.60 ± 01.95	$\mathbf{513.22 \pm 00.82}$
Cartpole Balance Sparse	$\mathbf{491.07 \pm 00.00}$	485.34 ± 49.39
Cartpole Swingup	347.58 ± 18.30	$\mathbf{371.24 \pm 05.62}$
Cartpole Swingup Sparse	36.98 ± 42.83	$\mathbf{102.44 \pm 139.79}$
Cheetah Run	$\mathbf{315.40 \pm 39.64}$	305.91 ± 103.62
Cup Catch	465.23 ± 28.77	$\mathbf{468.32 \pm 12.53}$
Finger Spin	186.66 ± 39.34	$\mathbf{245.77 \pm 61.99}$
Finger Turn Easy	359.32 ± 76.13	$\mathbf{414.82 \pm 46.09}$
Finger Turn Hard	347.91 ± 81.80	$\mathbf{398.38 \pm 63.40}$
Hopper Hop	91.05 ± 29.62	$\mathbf{97.50 \pm 29.83}$
Hopper Stand	350.77 ± 88.92	$\mathbf{384.44 \pm 74.34}$
Pendulum Swingup	$\mathbf{301.01 \pm 39.80}$	233.66 ± 199.66
Reacher Easy	463.30 ± 17.78	$\mathbf{477.51 \pm 14.02}$
Reacher Hard	391.94 ± 148.99	$\mathbf{466.35 \pm 25.94}$
Walker Run	238.07 ± 76.42	$\mathbf{271.72 \pm 63.37}$
Walker Stand	$\mathbf{462.81 \pm 18.20}$	460.51 ± 41.87
Walker Walk	429.65 ± 27.06	$\mathbf{440.85 \pm 49.87}$

Table 5.2: Average final return and standard deviation over five trainings in the Velocity Control environments.

Task	Uninformed	Informed
Concentration	$\mathbf{00.01 \pm 00.16}$	-0.24 ± 00.09
Count Recall	-0.66 ± 00.17	$\mathbf{-0.58 \pm 00.24}$
Higher Lower	$\mathbf{00.39 \pm 00.07}$	00.31 ± 00.12
Mine Sweeper	$\mathbf{-0.06 \pm 00.32}$	-0.07 ± 00.38
Noisy Position Cart Pole	00.21 ± 00.19	$\mathbf{00.23 \pm 00.27}$
Noisy Position Pendulum	00.54 ± 00.06	$\mathbf{00.55 \pm 00.05}$
Position Cart Pole	00.75 ± 00.00	$\mathbf{00.75 \pm 00.00}$
Position Pendulum	00.64 ± 00.07	$\mathbf{00.65 \pm 00.04}$
Repeat First	00.24 ± 00.87	$\mathbf{00.56 \pm 01.00}$
Repeat Previous	-0.01 ± 00.18	$\mathbf{00.44 \pm 00.13}$

Table 5.3: Average final return and standard deviation over five trainings in the Pop Gym environments.

time dependencies, such as the Repeat Previous environment of the Pop Gym suite. In this subsection, we study in depth this failure case of the Uninformed Dreamer for this particular environment. In the Repeat Previous environment, the agent is observing random noise, and is rewarded for outputting the observation that it got k time steps ago. While in Subsection 5.6.3 we only considered the default Pop Gym environments, where $k = 4$ for the Repeat Previous environment, we here consider the Medium ($k = 32$) and Hard ($k = 64$) versions of this environment.

In Figure 5.8, we see that the Uninformed Dreamer is not able to improve the performance of its policy at all in these harder environments, while the Informed Dreamer still seems to converge towards a near-optimal policy. It once

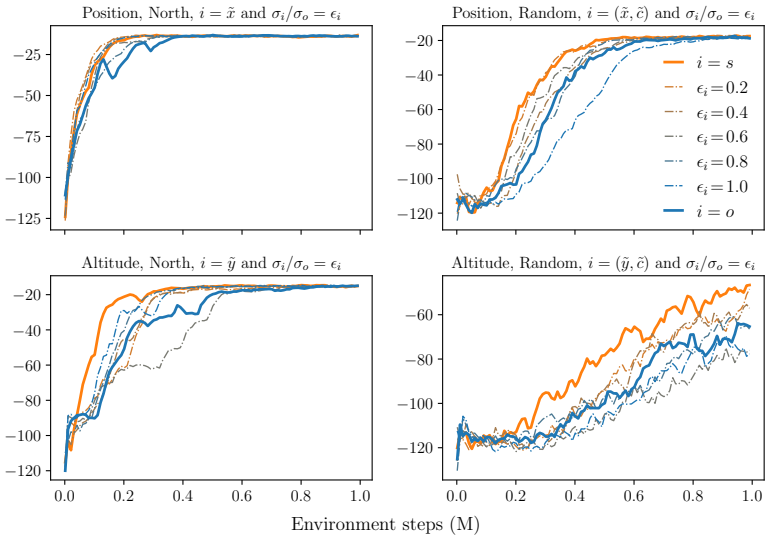


Figure 5.7: Varying Mountain Hike environments: average return of the Informed Dreamer with various level of information over five trainings.

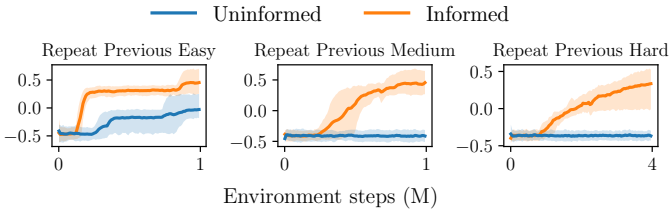


Figure 5.8: Uninformed Dreamer and Informed Dreamer with $i = s$ in the Repeat Previous environments: minimum, maximum and average returns over five trainings.

again validates empirically the assumption that exploiting additional information about the state improves the speed of convergence towards an optimal policy. Even more, it shows that exploiting additional information about the state can lead to convergence in environments where traditional approaches fail, such as those with long time dependencies. The additional supervision provided by this Markovian information (the last k observations) certainly endows the statistic $z \sim f(\cdot|h)$ with a useful encoding of the last k observations, which is then decoded by the policy. Table 5.4 provides the final return obtained by the Uninformed Dreamer and the Informed Dreamer for these environments.

5.E.3 Flickering Atari

In the Flickering Atari environments, the agent is tasked with playing the Atari games [Bellemare et al., 2013] on a flickering screen. The dynamics are left unchanged, but the agent may randomly observe a blank screen instead of the game screen, with probability $p = 0.5$. While the classic Atari games are known

Task	Uninformed	Informed
Repeat Previous Easy	-0.01 ± 00.18	$\mathbf{00.44 \pm 00.13}$
Repeat Previous Medium	-0.41 ± 00.06	$\mathbf{00.46 \pm 00.16}$
Repeat Previous Hard	-0.36 ± 00.07	$\mathbf{00.33 \pm 00.19}$

Table 5.4: Average final return and standard deviation over five trainings in the Repeat Previous environments.

to have low stochasticity and few partial observability challenges [Hausknecht and Stone, 2015], their flickering counterparts have constituted a classic benchmark in the partially observable RL literature [Hausknecht and Stone, 2015, Zhu et al., 2017, Igl et al., 2018, Ma et al., 2020]. Moreover, regarding the recent advances in sample-efficiency of model-based RL approaches, we consider the Atari 100k benchmark, where only 100k actions can be taken by the agent for generating samples of interaction.

For these environments, we consider the RAM state of the simulator, a 128-dimensional byte vector, to be available as additional information for supervision. This information vector is indeed guaranteed to satisfy the conditional independence of the informed POMDP: $p(o|i, s) = p(o|i)$. Moreover, we post-process this additional information by only selecting the subset of variables that are relevant to the game that is considered, according to the annotations provided by Anand et al. [2019]. Depending on the game, this information vector might contain the number of remaining opponents, their positions, the player position, etc.

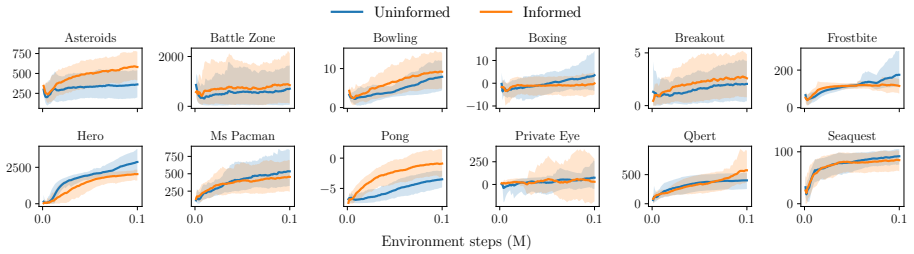


Figure 5.9: Uninformed Dreamer and Informed Dreamer with $i = \phi(\text{RAM})$ in the Flickering Atari environments: minimum, maximum and average returns over five trainings.

Figure 5.9 shows that the speed of convergence and the performance of the policies is greatly improved by considering additional information for six environments, while degraded for four others and left similar for the rest. The final returns are given in Table 5.5, offering similar conclusions.

5.E.4 Flickering Control

In the Flickering Control environments, the agent performs one of the standard DeepMind Control tasks from images but through a flickering screen. As with the Flickering Atari environments, the dynamics are left unchanged, except

Task	Uninformed	Informed
Asteroids	362.17 ± 112.95	$\mathbf{580.92 \pm 95.61}$
Battle Zone	706.67 ± 776.00	$\mathbf{849.61 \pm 357.35}$
Bowling	07.89 ± 02.00	$\mathbf{09.17 \pm 01.24}$
Boxing	$\mathbf{03.54 \pm 12.33}$	-0.06 ± 05.66
Breakout	02.06 ± 01.32	$\mathbf{02.59 \pm 01.47}$
Frostbite	$\mathbf{174.96 \pm 84.31}$	115.43 ± 30.20
Hero	$\mathbf{2864.66 \pm 1054.84}$	2033.51 ± 226.50
Ms Pacman	$\mathbf{534.67 \pm 117.97}$	455.02 ± 155.17
Pong	-3.49 ± 01.19	$\mathbf{-0.90 \pm 01.78}$
Private Eye	$\mathbf{74.27 \pm 42.00}$	29.66 ± 67.47
Qbert	401.27 ± 117.26	$\mathbf{574.70 \pm 26.92}$
Seaquest	$\mathbf{91.44 \pm 13.60}$	83.95 ± 21.11

Table 5.5: Average final return and standard deviation over five trainings in the Flickering Atari environments.

that the agent may randomly observe a blank screen instead of the task screen, with probability $p = 0.5$. For these environments, we consider the state to be available as additional information, as for the Velocity Control environments.

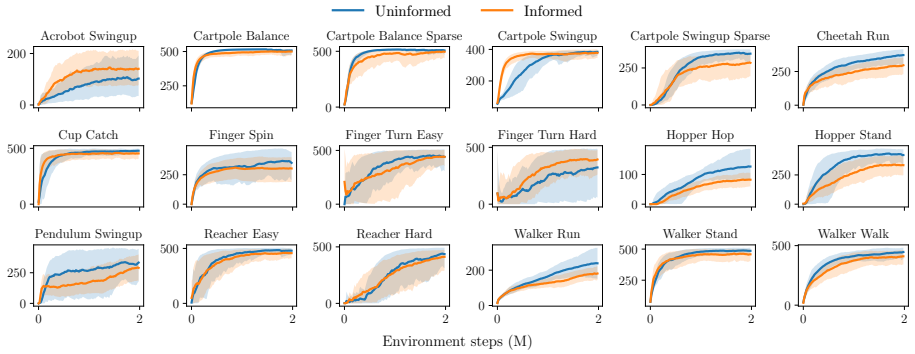


Figure 5.10: Uninformed Dreamer and Informed Dreamer with $i = s$ in the Flickering Control environments: minimum, maximum and average returns over five trainings.

Regarding this benchmark, considering additional information seems to degrade learning, generally resulting in worse policies. This suggests that not all information is good to learn, some might be irrelevant to the control task and hinders the learning of optimal policies. The final returns are given in Table 5.6, and offer similar conclusions.

Task	Uninformed	Informed
Acrobot Swingup	104.87 ± 54.88	$\mathbf{141.49 \pm 72.53}$
Cartpole Balance	$\mathbf{508.01 \pm 00.92}$	499.95 ± 24.87
Cartpole Balance Sparse	$\mathbf{507.94 \pm 03.04}$	495.14 ± 69.63
Cartpole Swingup	$\mathbf{384.37 \pm 14.66}$	377.60 ± 32.62
Cartpole Swingup Sparse	$\mathbf{347.07 \pm 27.63}$	284.53 ± 72.05
Cheetah Run	$\mathbf{372.96 \pm 30.98}$	296.70 ± 23.34
Cup Catch	$\mathbf{478.61 \pm 12.53}$	455.59 ± 13.58
Finger Spin	$\mathbf{349.85 \pm 123.88}$	303.03 ± 76.30
Finger Turn Easy	$\mathbf{441.53 \pm 47.13}$	441.16 ± 66.91
Finger Turn Hard	323.19 ± 200.67	$\mathbf{392.48 \pm 85.25}$
Hopper Hop	$\mathbf{126.72 \pm 37.89}$	81.92 ± 19.90
Hopper Stand	$\mathbf{420.38 \pm 57.48}$	331.48 ± 27.61
Pendulum Swingup	$\mathbf{329.35 \pm 82.31}$	286.53 ± 102.18
Reacher Easy	$\mathbf{479.25 \pm 18.15}$	457.72 ± 19.31
Reacher Hard	$\mathbf{433.40 \pm 214.42}$	412.97 ± 27.10
Walker Run	$\mathbf{239.22 \pm 92.40}$	180.63 ± 27.73
Walker Stand	$\mathbf{485.78 \pm 46.26}$	457.36 ± 37.65
Walker Walk	$\mathbf{447.03 \pm 26.83}$	409.72 ± 68.67

Table 5.6: Average final return and standard deviation over five trainings in the Flickering Control environments.

Chapter 6

Learning Faster with Additional Information

A Theoretical Justification for Asymmetric Actor-Critic Algorithms. Gaspard Lambrechts, Damien Ernst and Aditya Mahajan.

From the paper to appear at the *International Conference on Machine Learning.*

Abstract

In reinforcement learning for partially observable environments, many successful algorithms have been developed within the asymmetric learning paradigm. This paradigm leverages additional state information available at training time for faster learning. Although the proposed learning objectives are usually theoretically sound, these methods still lack a precise theoretical justification for their potential benefits. We propose such a justification for asymmetric actor-critic algorithms with linear function approximators by adapting a finite-time convergence analysis to this setting. The resulting finite-time bound reveals that the asymmetric critic eliminates error terms arising from aliasing in the agent state.

6.1 Introduction

Reinforcement learning (RL) is an appealing framework for solving decision making problems, notably because it makes very few assumptions about the problem at hand. In its purest form, the promise of an RL algorithm is to learn an optimal behavior from interaction with an environment whose dynamics are unknown. More formally, an RL algorithm aims to learn a policy – which is defined as a mapping from observations to actions – from interaction samples, in order to maximize a reward signal. While RL has obtained empirical successes for a plethora of challenging problems ranging from games to robotics [Mnih et al., 2015, Schrittwieser et al., 2020, Levine et al., 2015, Akkaya et al., 2019], most of these achievements have assumed full state observability. A more realistic assumption is partial state observability, where only a partial observation of the state of the environment is available for taking actions. In this setting, the optimal action generally depends on the complete history of past observations and actions. Traditional RL approaches have thus been adapted by considering history-dependent policies, usually with a recurrent neural network to process histories [Bakker, 2001, Wierstra et al., 2007, Hausknecht and Stone, 2015, Heess et al., 2015, Zhang et al., 2016, Zhu et al., 2017]. Given the difficulty of learning effective history-dependent policies, various auxiliary representation learning objectives have been proposed to compress the history into useful representations [Igl et al., 2018, Buesing et al., 2018, Guo et al., 2018, Gregor et al., 2019, Han et al., 2019, Guo et al., 2020, Lee et al., 2020, Subramanian et al., 2022, Ni et al., 2024]. Such methods usually seek to learn history representations that encode the belief, defined as the posterior distributions over the states given the history, which is a sufficient statistic of the history for optimal control.

While these methods are theoretically able to learn optimal history-dependent policies, they usually learn solely from the partial state observations, which can be restrictive. Indeed, assuming the same partial observability at training time and execution time can be too pessimistic for many environments, notably for those that are simulated. This motivated the asymmetric learning paradigm, where additional state information available at training time is leveraged during the process of learning a history-dependent policy. Although the optimal policies obtained by asymmetric learning are theoretically equivalent to those learned by symmetric learning, the promise of asymmetric learning is to improve the convergence speed. Early approaches proposed to imitate a privileged policy conditioned on the state [Choudhury et al., 2018], or to use an asymmetric critic conditioned on the state [Pinto et al., 2018]. These heuristic methods initially lacked a theoretical framework, and a recent line of work has focused on proposing theoretically grounded asymmetric learning objectives. First, imitation learning of a privileged policy was known to be suboptimal, and it was addressed by constraining the privileged policy so that its imitation results in an optimal policy for the partially observable environment [Warrington et al., 2021]. Similarly, asymmetric actor-critic approaches were proven to provide biased gradients, and an unbiased actor-critic approach was proposed by introducing the history-state value function [Baisero and Amato, 2022]. In model-based RL, several works proposed world model objectives that are proved to provide sufficient statistics of the history, by leveraging the state [Avalos et al., 2024]

or arbitrary state information [Lambrechts et al., 2024a]. Finally, asymmetric representation learning approaches were proposed to learn sufficient statistics from state samples [Wang et al., 2023, Sinha and Mahajan, 2023]. It is worth noting that many recent successful applications of RL have greatly benefited from asymmetric learning, usually through an asymmetric critic [Degrave et al., 2022, Kaufmann et al., 2023, Vasco et al., 2024].

Despite these methods being theoretically grounded, in the sense that policies satisfying these objectives are optimal policies, they still lack a theoretical justification for their potential benefit. In particular, there is no theoretical justification for the improved convergence speed of asymmetric learning. In this work, we propose such a justification for an asymmetric actor-critic algorithm, using agent-state policies and linear function approximators. Agent-state policies rely on an internal state, which is updated recurrently based on successive actions and observations, from which the next action is selected. This agent state can introduce aliasing, a phenomenon in which an agent state may correspond to two different beliefs. Our argument relies on the comparaison of two analogous finite-time bounds: one for a symmetric natural actor-critic algorithm [Cayci et al., 2024], and its adaptation to the asymmetric setting that we derive in this paper. This comparison reveals that asymmetric learning eliminates error terms arising from aliasing in the agent state in symmetric learning. These aliasing terms are given by the difference between the true belief (i.e., the posterior distribution over the states given the history) and the approximate belief (i.e., the posterior distribution over the states given the agent state). This suggests that asymmetric learning may be particularly useful when aliasing is high.

A recent related work proposed a model-based asymmetric actor-critic algorithm relying on belief approximation, and proved its sample efficiency [Cai et al., 2024]. It also considered agent-state policies, and studied the finite-time performance by providing a probably approximately correct (PAC) bound, instead of an expectation bound as here. While the algorithm was restricted to finite horizon and discrete spaces, notably for implementing count-based exploration strategies, it tackled the online exploration setting and its performance bound did not present a concentrability coefficient. This related analysis thus provides a promising framework for future works in a more challenging setting. However, it did not study the existing asymmetric actor-critic algorithm, and did not provides a direct comparison with symmetric learning. In contrast, we focus on providing comparable bounds for the existing model-free asymmetric actor-critic algorithm and its symmetric counterpart.

In Section 6.2, we formalize the environments, policies, and Q-functions that are considered. In Section 6.3, we introduce the asymmetric and symmetric actor-critic algorithms that are studied. In Section 6.4, we provide the finite-time bounds for the asymmetric and symmetric actor-critic algorithms. Finally, in Section 6.5, we conclude by summarizing the contributions and providing avenues for future works.

6.2 Background

In Subsection 6.2.1, we introduce the decision processes and agent-state policies that are considered. Then, we introduce the asymmetric and symmetric Q-function for such policies, in Subsection 6.2.2 and Subsection 6.2.3, respectively.

6.2.1 Partially Observable Markov Decision Process

A partially observable Markov decision process (POMDP) is a tuple $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, P, T, R, O, \gamma)$, with discrete state space $\mathcal{S}$, discrete action space $\mathcal{A}$, and discrete observation space $\mathcal{O}$. The initial state distribution P gives the probability $P(s_0)$ of $s_0 \in \mathcal{S}$ being the initial state of the decision process. The dynamics are described by the transition distribution T that gives the probability $T(s_{t+1}|s_t, a_t)$ of $s_{t+1} \in \mathcal{S}$ being the state resulting from action $a_t \in \mathcal{A}$ in state $s_t \in \mathcal{S}$. The reward function R gives the immediate reward $r_t = R(s_t, a_t, s_{t+1})$ of the reward $r_t \in [0, 1]$ resulting from this transition. The observation distribution O gives the probability $O(o_t|s_t)$ to get observation $o_t \in \mathcal{O}$ in state $s_t \in \mathcal{S}$. Finally, the discount factor $\gamma \in [0, 1)$ weights the relative importance of future rewards. Taking a sequence of t actions in the POMDP conditions its execution and provides the history $h_t = (o_0, a_0, \dots, o_t) \in \mathcal{H}$, where $\mathcal{H}$ is the set of histories of arbitrary length. In general, the optimal policy in a POMDP depends on the complete history.

However, in practice it is infeasible to learn a policy conditioned on the full history, since the latter grows unboundedly with time. We consider an agent-state policy $\pi \in \Pi_{\mathcal{M}}$ that uses an agent-state process $\mathcal{M} = (\mathcal{Z}, U)$, in order to take actions Dong et al. [2022], Sinha and Mahajan [2024]. More formally, we consider a discrete agent state space $\mathcal{Z}$, and an update distribution U that gives the probability $U(z_{t+1}|z_t, a_t, o_{t+1})$ of $z_{t+1} \in \mathcal{Z}$ being the state resulting from action $a_t \in \mathcal{A}$ and observation $o_{t+1} \in \mathcal{O}$ in agent state $z_t \in \mathcal{Z}$. Note that the update distribution U also describe the initial agent state distribution with $z_{-1} \notin \mathcal{Z}$ the null agent state and $a_{-1} \notin \mathcal{A}$ the null action. Some examples of agent states that are often used are a sliding window of past observations, or a belief filter. Aliasing may occur when the agent state does not summarize all information from the history about the state of the environment, see Appendix 6.A for an example. Given the agent state z_t, the policy π samples actions according to $a_t \sim \pi(\cdot|z_t)$. An agent-state policy $\pi^* \in \Pi_{\mathcal{M}}$ is said to be optimal for an agent-state process $\mathcal{M}$ if it maximizes the expected discounted sum of rewards: $\pi^* \in \arg\max_{\pi \in \Pi_{\mathcal{M}}} J(\pi)$ with $J(\pi) = \mathbb{E}^{\pi}[\sum_{t=0}^{\infty} \gamma^t R_t]$.

In the following, we denote by S_t, O_t, Z_t, A_t and R_t the random variables induced by the POMDP $\mathcal{P}$. Given a POMDP $\mathcal{P}$ and an agent-state process $\mathcal{M}$, the initial environment-agent state distribution P is given by,

$$P(s_0, z_0) = P(s_0) \sum_{o_0 \in \mathcal{O}} O(o_0|s_0) U(z_0|z_{-1}, a_{-1}, o_0). \tag{6.1}$$

Furthermore, given an agent-state policy $\pi \in \Pi_{\mathcal{M}}$, we define the discounted

visitation distribution as,

$$d^\pi(s,z) = (1-\gamma)\sum_{s_0,z_0} P(s_0,z_0)\sum_{t=0}^{\infty}\gamma^k \Pr(S_t = s, Z_t = z | S_0 = s_0, Z_0 = z_0). \tag{6.2}$$

Finally, we define the visitation distribution m steps from the discounted visitation distribution as,

$$d^\pi_m(s,z) = \sum_{s_0,z_0} d^\pi(s_0,z_0)\Pr(S_m = s, Z_m = z | S_0 = s_0, Z_0 = z_0). \tag{6.3}$$

In the following, we define the various value functions for the policies that we defined. Note that we use calligraphic letters $\mathcal{Q}^\pi$, $\mathcal{V}^\pi$ and $\mathcal{A}^\pi$ for the asymmetric functions, and regular letters Q^π, V^π and A^π for the symmetric ones.

6.2.2 Asymmetric Q-function

Similarly to the asymmetric Q-function of Baisero and Amato [2022], which is conditioned on (s,h,a), we define an asymmetric Q-function that we condition on (s,z,a), where z is the agent state resulting from history h. The asymmetric Q-function $\mathcal{Q}^\pi$ of an agent-state policy $\pi \in \Pi_{\mathcal{M}}$ is defined as the expected discounted sum of rewards, starting from environment state s, agent state z, and action a, and using policy π afterwards,

$$\mathcal{Q}^\pi(s,z,a) = \mathbb{E}^\pi\left[\sum_{t=0}^{\infty}\gamma^t R_t \middle| S_0 = s, Z_0 = z, A_0 = a\right]. \tag{6.4}$$

The asymmetric value function $\mathcal{V}^\pi$ of an agent-state policy $\pi \in \Pi_{\mathcal{M}}$ is defined as $\mathcal{V}^\pi(s,z) = \sum_{a\in\mathcal{A}}\pi(a|z)\mathcal{Q}^\pi(s,z,a)$. We also define the asymmetric advantage function $\mathcal{A}^\pi(s,z,a) = \mathcal{Q}^\pi(s,z,a) - \mathcal{V}^\pi(s,z)$.

Let us define the m-step asymmetric Bellman operator as,

$$\widetilde{\mathcal{Q}}^\pi(s,z,a) = \mathbb{E}^\pi\left[\sum_{t=0}^{m-1}\gamma^t R_t + \gamma^m \widetilde{\mathcal{Q}}^\pi(S_m, Z_m, A_m) \;\middle|\; S_0 = s, Z_0 = z, A_0 = a\right]. \tag{6.5}$$

Since this m-step asymmetric Bellman operator is γ^m-contractive, equation (6.5) has a unique fixed point $\widetilde{\mathcal{Q}}^\pi$. Notice that, when using an agent-state policy, the environment state and agent state (S_t, Z_t) are Markovian. Therefore, it can be shown that the fixed point $\widetilde{\mathcal{Q}}^\pi$ is the same as the asymmetric Q-function $\mathcal{Q}^\pi$.

6.2.3 Symmetric Q-function

The symmetric Q-function Q^π of an agent-state policy $\pi \in \Pi_{\mathcal{M}}$ in a POMDP $\mathcal{P}$ is defined as the expected discounted sum of rewards, starting from agent state z and action a, and using policy π afterwards,

$$Q^\pi(z,a) = \mathbb{E}^\pi\left[\sum_{t=0}^{\infty}\gamma^t R_t \middle| Z_0 = z, A_0 = a\right]. \tag{6.6}$$

The symmetric value function V^π of an agent-state policy $\pi \in \Pi_{\mathcal{M}}$ is defined as $V^\pi(z) = \sum_{a \in \mathcal{A}} \pi(a|z) Q^\pi(z,a)$. We also define the symmetric advantage function $A^\pi(z,a) = Q^\pi(z,a) - V^\pi(z)$.

Let us define the m-step symmetric Bellman operator as,

$$\widetilde{Q}^\pi(z,a) = \mathbb{E}^\pi \left[\sum_{t=0}^{m-1} \gamma^t R_t + \gamma^m \widetilde{Q}^\pi(Z_m, A_m) \middle| Z_0 = z, A_0 = a \right]. \tag{6.7}$$

It can be verified that the m-step symmetric Bellman operator is γ^m-contractive. Therefore, equation (6.7) has a unique fixed point $\widetilde{Q}^\pi$. However, because the agent state is not necessarily Markovian, in general $Q^\pi \neq \widetilde{Q}^\pi$.

6.3 Natural Actor-Critic Algorithms

In this section, we present the asymmetric and symmetric natural actor-critic algorithms, which make use of an actor, or policy, and a critic, or Q-function. The asymmetric variant will use an asymmetric critic, learned using asymmetric temporal difference learning, while the symmetric variant will use a symmetric critic, learned using symmetric temporal difference learning. These temporal difference learning algorithms are presented in in Subsection 6.3.1 and Subsection 6.3.2, respectively. Then, Subsection 6.3.3 presents the complete natural actor-critic algorithm that uses a temporal difference learning algorithm as a subroutine.

For any Euclidean space $\mathcal{X}$, let $\mathcal{B}_2(0,B)$ be the ℓ_2-ball centered at the origin with radius $B > 0$, and let $\Gamma_{\mathcal{C}} \colon \mathcal{X} \to \mathcal{C}$ be a projection operator into the closed and convex set $\mathcal{C} \subseteq \mathcal{X}$ in ℓ_2-norm: $\Gamma_{\mathcal{C}}(x) \in \arg\min_{c \in \mathcal{C}} \|c - x\|_2^2 \subseteq \mathcal{C},\ \forall x \in \mathcal{X}$. Finally, let us define the μ-weighted ℓ_2-norm, for any probability measures $\mu \in \Delta(\mathcal{X})$ as,

$$\|f\|_\mu = \sqrt{\sum_{x \in \mathcal{X}} \mu(x) \left|f(x)\right|^2}. \tag{6.8}$$

In the algorithms, we implicitly assume to be able to directly sample from the discounted visitation measure d^π. When it is unrealistic, it is still possible to sample from d^π by sampling an initial time step $t_0 \sim \text{Geom}(1-\gamma)$ from a geometric distribution with success rate $1-\gamma$, and then taking $t_0 - 1$ actions in the POMDP. The resulting sample (s_{t_0}, z_{t_0}) follows the distribution d^π.

6.3.1 Asymmetric Critic

Suppose we are given features $\phi \colon \mathcal{S} \times \mathcal{Z} \times \mathcal{A} \to \mathbb{R}^{d_\phi}$. Without loss of generality, we assume $\sup_{s,z,a} \|\phi(s,z,a)\|_2 \leq 1$. Given a weight vector $\beta \in \mathbb{R}^{d_\phi}$, let $\widehat{\mathcal{Q}}^\pi_\beta$ denote the linear approximation of the asymmetric Q-function $\mathcal{Q}^\pi$ that uses features ϕ with weight β,

$$\widehat{\mathcal{Q}}^\pi_\beta(s,z,a) = \langle \beta, \phi(s,z,a) \rangle. \tag{6.9}$$

Given an arbitrary projection radius $B > 0$, we define the hypothesis space as,

$$\mathcal{F}^B_\phi = \{(s,z,a) \mapsto \langle \beta, \phi(s,z,a) \rangle : \beta \in \mathcal{B}_2(0,B)\}. \tag{6.10}$$

We denote the optimal parameter of the asymmetric critic approximation by $\beta_*^\pi \in \arg\min_{\beta \in \mathcal{B}_2(0,B)} \|\langle \beta, \phi(\cdot)\rangle - \mathcal{Q}^\pi(\cdot)\|_d$, and denote the corresponding approximation by $\widehat{\mathcal{Q}}_*^\pi(\cdot) = \langle \beta_*^\pi, \phi(\cdot)\rangle$. The corresponding error is,

$$\varepsilon_{\text{app}} = \min_{f \in \mathcal{F}_\phi^B} \left\| f - \mathcal{Q}^\pi \right\|_d = \left\| \widehat{\mathcal{Q}}_*^\pi - \mathcal{Q}^\pi \right\|_d, \tag{6.11}$$

with $d(s, z, a) = d^\pi(s, z)\pi(a|z)$ the sampling distribution.

In Algorithm 6.1, we present the m-step temporal difference learning algorithm for approximating the asymmetric Q-function $\mathcal{Q}^\pi$ of an arbitrary agent-state policy $\pi \in \Pi_{\mathcal{M}}$. At each step k, the algorithm obtains one sample $(s_{k,0}, z_{k,0}) \sim d^\pi$ from the discounted visitation distribution. Then, m actions are selected according to policy π to provide samples $(a_{k,t}, r_{k,t}, s_{k,t+1}, o_{k,t+1}, z_{k,t+1})$ for $0 \leq t < m$. Next, the temporal difference δ_k and semi-gradient g_k are computed, based on a last action $a_{k,m} \sim \pi(\cdot|z_{k,m})$,

$$\delta_k = \sum_{i=0}^{m-1} \gamma^i r_{k,i} + \gamma^m \widehat{\mathcal{Q}}_{\beta_k}^\pi(s_{k,m}, z_{k,m}, a_{k,m}) - \widehat{\mathcal{Q}}_{\beta_k}^\pi(s_{k,0}, z_{k,0}, a_{k,0}), \tag{6.12}$$

$$g_k = \delta_k \nabla_\beta \widehat{\mathcal{Q}}_{\beta_k}^\pi(s_{k,0}, z_{k,0}, a_{k,0}). \tag{6.13}$$

Then, the semi-gradient update is performed with $\beta_{k+1}^- = \beta_k + \alpha g_k$ and the parameters are projected onto the ball of radius B: $\beta_{k+1} = \Gamma_{\mathcal{B}_2(0,B)}(\beta_{k+1}^-)$. At the end, the algorithm computes the average parameter $\bar{\beta} = \frac{1}{K}\sum_{k=0}^{K-1} \beta_k$ and returns the average approximation $\overline{\mathcal{Q}}^\pi = \widehat{\mathcal{Q}}_{\bar{\beta}}^\pi$.

Algorithm 6.1: Multistep temporal difference learning algorithm.

parameters: m the bootstrap time step,
α the step size,
K the number of updates,
B the projection radius.

inputs: $\pi \in \Pi_{\mathcal{M}}$ the policy.

for $k = 0 \dots K-1$ **do**
  Initialize $(s_{k,0}, z_{k,0}) \sim d^\pi$.
  for $i = 0 \dots, m-1$ **do**
    Select action $a_{k,i} \sim \pi(\cdot|z_{k,i})$.
    Get environment state $s_{k,i+1} \sim T(\cdot|s_{k,i}, a_{k,i})$.
    Get reward $r_{k,i} = R(s_{k,i}, a_{k,i}, s_{k,i+1})$.
    Get observation $o_{k,i+1} \sim O(\cdot|s_{k,i+1})$.
    Update agent state $z_{k,i+1} \sim U(\cdot|z_{k,i}, a_{k,i}, o_{k,i+1})$.
  Sample last action $a_{k,m} \sim \pi(\cdot|z_{k,m})$.
  Compute semi-gradient g_k according to equation (6.13) or equation (6.17).
  Update $\beta_{k+1} = \Gamma_{\mathcal{B}_2(0,B)}(\beta_k + \alpha g_k)$.
Compute average parameter $\bar{\beta} = \frac{1}{K}\sum_{k=0}^{K-1} \beta_k$.
return *average estimate* $\overline{\mathcal{Q}}^\pi(\cdot) = \widehat{\mathcal{Q}}_{\bar{\beta}}^\pi(\cdot) = \langle \bar{\beta}, \phi(\cdot)\rangle$ *or* $\overline{Q}^\pi(\cdot) = \widehat{Q}_{\bar{\beta}}^\pi(\cdot) = \langle \bar{\beta}, \chi(\cdot)\rangle$.

6.3.2 Symmetric Critic

Similarly, we suppose that we are given features $\chi \colon \mathcal{Z} \times \mathcal{A} \to \mathbb{R}^{d_\chi}$. Without loss of generality, we assume $\sup_{z,a} \|\chi(z, a)\|_2 \leq 1$. Given a weight vector $\beta \in \mathbb{R}^{d_\chi}$,

let $\widehat{Q}^{\pi}_{\beta}$ denote the linear approximation of the symmetric Q-function Q^{π} that uses features χ with weight β,

$$\widehat{Q}^{\pi}_{\beta}(z,a) = \langle \beta, \chi(z,a) \rangle. \tag{6.14}$$

The corresponding hypothesis space for an arbitrary projection radius $B > 0$ is denoted with $\mathcal{F}^{B}_{\chi}$. The optimal parameter is also denoted by $\beta^{\pi}_{*} \in \arg\min_{\beta \in \mathcal{B}_2(0,B)} \|\langle \beta, \chi(\cdot) \rangle - Q^{\pi}(\cdot)\|_d$, the corresponding optimal approximation is $\widehat{Q}^{\pi}_{*} = \langle \beta^{\pi}_{*}, \chi(\cdot) \rangle$, and the corresponding error is,

$$\varepsilon_{\text{app}} = \min_{f \in \mathcal{F}^{B}_{\chi}} \left\| f - Q^{\pi} \right\|_d = \left\| \widehat{Q}^{\pi}_{*} - Q^{\pi} \right\|_d, \tag{6.15}$$

with $d(z,a) = \sum_{s \in \mathcal{S}} d^{\pi}(s,z)\pi(a|z)$ the sampling distribution.

Algorithm 6.1 also presents the m-step temporal difference learning algorithm for approximating the symmetric Q-function. The latter is identical to that of the asymmetric Q-function except that states are not exploited, such that the temporal difference δ_k and semi-gradient g_k are given by,

$$\delta_k = \sum_{i=0}^{m-1} \gamma^i r_{k,i} + \gamma^m \widehat{Q}^{\pi}_{\beta_k}(z_{k,m}, a_{k,m}) - \widehat{Q}^{\pi}_{\beta_k}(z_{k,0}, a_{k,0}), \tag{6.16}$$

$$g_k = \delta_k \nabla_{\beta} \widehat{Q}^{\pi}_{\beta_k}(z_{k,0}, a_{k,0}). \tag{6.17}$$

At the end, the algorithm returns the average symmetric approximation $\overline{Q}^{\pi} = \widehat{Q}^{\pi}_{\bar{\beta}}$. Note that this symmetric critic approximation and temporal difference learning algorithm corresponds to the one proposed by Cayci et al. [2024].

6.3.3 Natural Actor-Critic Algorithms

For both the asymmetric and symmetric actor-critic algorithms, we consider a log-linear agent-state policy $\pi_{\theta} \in \Pi_{\mathcal{M}}$. More precisely, the policy uses features $\psi \colon \mathcal{Z} \times \mathcal{A} \to \mathbb{R}^{d_{\psi}}$, with $\sup_{z,a} \|\psi(z,a)\|_2 \leq 1$ without loss of generality, and a softmax readout,

$$\pi_{\theta}(a_t|z_t) = \frac{\exp(\langle \theta, \psi(z_t, a_t) \rangle)}{\sum_{a \in \mathcal{A}} \exp(\langle \theta, \psi(z_t, a) \rangle)}. \tag{6.18}$$

In this work, we consider natural policy gradients, which are less sensitive to policy parametrization [Kakade, 2001]. Instead of computing the policy gradient in the original metric space, the idea is to compute the policy gradient on a statistical manifold, defined by the expected Fisher information metric. The natural policy gradient is thus given by the standard policy gradient multiplied by a preconditioner Fisher information matrix. Natural policy gradients are at the core of many effective modern policy-gradient methods [Schulman et al., 2015].

The natural policy gradient of policy $\pi_{\theta} \in \Pi_{\mathcal{M}}$ is defined as follows [Kakade, 2001],

$$w^{\pi_{\theta}}_{*} = (1-\gamma) F^{\dagger}_{\pi_{\theta}} \nabla_{\theta} J(\pi_{\theta}), \tag{6.19}$$

where $F_{\pi_\theta}^\dagger$ is the pseudoinverse of the Fisher information matrix, which is defined as the outer product of the score of the policy,

$$F_{\pi_\theta} = \mathbb{E}^{d^{\pi_\theta}} [\nabla_\theta \log \pi_\theta(A|Z) \otimes \nabla_\theta \log \pi_\theta(A|Z)]. \tag{6.20}$$

As shown in Theorem 6.1, the natural policy gradient $w_*^{\pi_\theta}$ is the minimizer of the asymmetric objective (6.22).

Theorem 6.1 (Asymmetric natural policy gradient). For any POMDP $\mathcal{P}$ and any agent-state policy $\pi_\theta \in \Pi_{\mathcal{M}}$, we have,

$$w_*^{\pi_\theta} = (1-\gamma) F_{\pi_\theta}^\dagger \nabla_\theta J(\pi_\theta) \in \arg\min_{w \in \mathbb{R}^{d_\psi}} \mathcal{L}(w), \tag{6.21}$$

with,

$$\mathcal{L}(w) = \mathbb{E}^{d^{\pi_\theta}} \Big[\big(\langle \nabla_\theta \log \pi_\theta(A|Z), w \rangle - \mathcal{A}^{\pi_\theta}(S, Z, A) \big)^2 \Big]. \tag{6.22}$$

The proof is given in Appendix 6.B. In practice, since the asymmetric advantage function is unknown, the algorithm estimates the natural policy gradient by stochastic gradient descent of $\mathcal{L}(\omega)$ using the approximation $\overline{\mathcal{A}}^{\pi_\theta}(S, Z, A) = \overline{\mathcal{Q}}^{\pi_\theta}(S, Z, A) - \overline{\mathcal{V}}^{\pi_\theta}(S, Z)$ with $\overline{\mathcal{V}}^{\pi_\theta} = \sum_{a \in \mathcal{A}} \pi_\theta(a|Z) \overline{\mathcal{Q}}(S, Z, a)$.

Our natural actor-critic algorithm generalizes the one of Cayci et al. [2024] to the asymmetric setting and is detailed in Algorithm 6.2. For each policy gradient step $0 \le t < T$, the natural policy gradient $w_*^{\pi_t}$ is first estimated using N steps of stochastic gradient descent. At each natural policy gradient estimation step $0 \le n < N$, the algorithm samples an initial state $(s_{t,n}, z_{t,n}) \sim d^{\pi_t}$ from the discounted distribution d^{π_t} and an action $a_{t,n} \sim \pi_t(\cdot|z_{t,n})$ according to the policy $\pi_t = \pi_{\theta_t}$. Then, the gradient $v_{t,n}$ of the natural policy gradient estimate $w_{t,n}$ is computed with,

$$v_{t,n} = \nabla_w \big(\langle \nabla_\theta \log \pi_\theta(a_{t,n}|z_{t,n}), w_{t,n} \rangle - \overline{\mathcal{A}}^{\pi_\theta}(s_{t,n}, z_{t,n}, a_{t,n}) \big)^2, \tag{6.23}$$

The gradient step is performed with $w_{t,n+1}^- = w_{t,n} - \zeta v_{t,n}$ and the parameters are projected onto the ball of radius B: $w_{t,n+1} = \Gamma_{\mathcal{B}_2(0,B)}(w_{t,n+1}^-)$. Finally, the algorithm computes the average parameter $\bar{w}_t = \frac{1}{N} \sum_{n=0}^{N-1} w_{t,n}$ and performs the policy gradient step: $\theta_{t+1} = \theta_t + \eta \bar{w}_t$. After all policy gradient steps, the final policy is returned.

As shown in Theorem 6.2, the natural policy gradient $w_*^{\pi_\theta}$ is also the minimizer of the symmetric objective (6.25).

Theorem 6.2 (Symmetric natural policy gradient). For any POMDP $\mathcal{P}$ and any agent-state policy $\pi_\theta \in \Pi_{\mathcal{M}}$, we have,

$$w_*^{\pi_\theta} = (1-\gamma) F_{\pi_\theta}^\dagger \nabla_\theta J(\pi_\theta) \in \arg\min_{w \in \mathbb{R}^{d_\psi}} L(w), \tag{6.24}$$

with,

$$L(w) = \mathbb{E}^{d^{\pi_\theta}} \Big[\big(\langle \nabla_\theta \log \pi_\theta(A|Z), w \rangle - A^{\pi_\theta}(Z, A) \big)^2 \Big]. \tag{6.25}$$

Algorithm 6.2: Natural actor-critic algorithm.

parameters: T the number of updates,
N number of gradient estimation steps,
ζ the gradient estimation step size,
η the step size,
B the projection radius.

Initialize $\theta_0 = 0$.
for $t = 0 \dots T-1$ **do**
  Obtain $\overline{\mathcal{Q}}^{\pi_t}$ or $\overline{Q}^{\pi_t}$ using Algorithm 6.1.
  Initialize $w_{t,0} = 0$.
  for $n = 0 \dots N-1$ **do**
    Initialize $(s_{t,n}, z_{t,n}) \sim d^{\pi_t}$.
    Sample $a_{t,n} \sim \pi_{\theta_t}(\cdot|z_{t,n})$.
    Compute the gradient $v_{t,n}$ of the policy gradient using equation (6.23) or equation (6.26).
    Update $w^-_{t,n+1} = w_{t,n} - \zeta v_{t,n}$.
    Project $w_{t,n+1} = \Gamma_{\mathcal{B}_2(0,B)}(w^-_{t,n+1})$.
  Update $\theta_{t+1} = \theta_t + \eta \frac{1}{N} \sum_{n=0}^{N-1} w_{t,n}$.
return *final policy* $\pi_T = \pi_{\theta_T}$.

The proof is given in Appendix 6.B. As in the asymmetric case, the symmetric advantage function is unknown, and the algorithm estimates the natural gradient by stochastic gradient descent of equation (6.25) using the approximation $\overline{A}^{\pi_\theta}(Z, A) = \overline{\mathcal{Q}}^{\pi_\theta}(Z, A) - \overline{V}^{\pi_\theta}(Z)$ with $\overline{V}^{\pi_\theta} = \sum_{a \in \mathcal{A}} \pi_\theta(a|Z) \overline{\mathcal{Q}}^{\pi_\theta}(Z, a)$.

Algorithm 6.2 also presents the symmetric natural actor-critic algorithm, initially proposed by Cayci et al. [2024]. The latter is similar to the asymmetric algorithm except that it uses the symmetric advantage function, such that the gradient of the policy gradient is given by,

$$v_{t,n} = \nabla_w \big(\langle \nabla_\theta \log \pi_\theta(a_{t,n}|z_{t,n}), w_{t,n} \rangle - \overline{A}^{\pi_\theta}(z_{t,n}, a_{t,n}) \big)^2. \tag{6.26}$$

While Theorem 6.1 and Theorem 6.2 show that $w_*^{\pi_\theta}$ is the minimizer of both the asymmetric and the symmetric objectives, the next section establishes the benefit of using the asymmetric loss. More precisely, asymmetric learning is shown to improve the estimation of the critic and thus the advantage function, which in turn results in a better estimation of the natural policy gradient.

6.4 Finite-Time Analysis

In this section, we give the finite-time bounds of the previous algorithms in both the asymmetric and symmetric cases. The bounds of the asymmetric and symmetric temporal difference learning algorithms are presented in Subsection 6.4.1 and Subsection 6.4.2, respectively. In Subsection 6.4.3, the bounds of the asymmetric and symmetric natural actor-critic algorithms are given.

We use $\|\mu - \nu\|_{\mathrm{TV}}$ to denote the total variation between two probability measures $\mu, \nu \in \Delta(\mathcal{X})$ over a discrete space $\mathcal{X}$,

$$\|\mu - \nu\|_{\mathrm{TV}} = \sup_{A \subseteq \mathcal{X}} |\mu(A) - \nu(A)| \tag{6.27}$$

$$= \frac{1}{2} \sum_{x \in \mathcal{X}} |\mu(x) - \nu(x)| \,. \tag{6.28}$$

6.4.1 Finite-Time Bound for the Asymmetric Critic

Our main result is to establish the following finite-time bound for the Q-function approximation resulting from the asymmetric temporal difference learning algorithm detailed in Algorithm 6.1.

Theorem 6.3 (Finite-time bound for asymmetric m-step temporal difference learning). For any agent-state policy $\pi \in \Pi_{\mathcal{M}}$, and any $m \in \mathbb{N}$, we have for Algorithm 6.1 with $\alpha = \frac{1}{\sqrt{K}}$ and arbitrary $B > 0$,

$$\sqrt{\mathbb{E}\left[\left\|\mathcal{Q}^\pi - \overline{\mathcal{Q}}^\pi\right\|_d^2\right]} \leq \varepsilon_{\text{td}} + \varepsilon_{\text{app}} + \varepsilon_{\text{shift}}, \tag{6.29}$$

where the temporal difference learning, function approximation, and distribution shift terms are given by,

$$\varepsilon_{\text{td}} = \sqrt{\frac{4B^2 + \left(\frac{1}{1-\gamma} + 2B\right)^2}{2\sqrt{K}(1-\gamma^m)}} \tag{6.30}$$

$$\varepsilon_{\text{app}} = \frac{1+\gamma^m}{1-\gamma^m} \min_{f \in \mathcal{F}_\phi^B} \|f - \mathcal{Q}^\pi\|_d \tag{6.31}$$

$$\varepsilon_{\text{shift}} = \left(B + \frac{1}{1-\gamma}\right) \sqrt{\frac{2\gamma^m}{1-\gamma^m} \sqrt{\|d_m - d\|_{\text{TV}}}}, \tag{6.32}$$

with $d(s,z,a) = d^\pi(s,z)\pi(a|z)$ the sampling distribution, and $d_m(s,z,a) = d_m^\pi(s,z)\pi(a|z)$ the bootstrapping distribution.

The proof is given in Appendix 6.C, and adapts the proof of Cayci et al. [2024] to the asymmetric setting. The first term ε_{td} is the usual temporal difference error term, decreasing in $K^{-1/4}$. The second term ε_{app} results from the use of linear function approximators. The third term $\varepsilon_{\text{shift}}$ arises from the distribution shift between the sampling distribution $d^\pi \otimes \pi$ (i.e., the discounted visitation measure) and the bootstrapping distribution $d_m^\pi \otimes \pi$ (i.e., the distribution m steps from the discounted visitation measure). It is a consequence of not assuming the existence of a stationary distribution nor assuming to sample from the stationary distribution.

6.4.2 Finite-Time Bound for the Symmetric Critic

Given a history $h_t = (o_0, a_0, \ldots, o_t)$, the belief is defined as,

$$b_t(s_t|h_t) = \Pr(S_t = s_t | H_t = h_t). \tag{6.33}$$

Given an agent state z_t, the approximate belief is defined as,

$$\hat{b}_t(s_t|z_t) = \Pr(S_t = s_t | Z_t = z_t). \tag{6.34}$$

We obtain the following finite-time bound for the Q-function approximation resulting from the symmetric temporal difference learning algorithm detailed in Algorithm 6.1.

Theorem 6.4 (Finite-time bound for symmetric m-step temporal difference learning [Cayci et al., 2024]). For any agent-state policy $\pi \in \Pi_{\mathcal{M}}$, and any $m \in \mathbb{N}$, we have for Algorithm 6.1 with $\alpha = \frac{1}{\sqrt{K}}$, and arbitrary $B > 0$,

$$\sqrt{\mathbb{E}\left[\left\|Q^\pi - \overline{Q}^\pi\right\|_d^2\right]} \leq \varepsilon_{\text{td}} + \varepsilon_{\text{app}} + \varepsilon_{\text{shift}} + \varepsilon_{\text{alias}}, \tag{6.35}$$

where the temporal difference learning, function approximation, distribution shift, and aliasing terms are given by,

$$\varepsilon_{\text{td}} = \sqrt{\frac{4B^2 + \left(\frac{1}{1-\gamma} + 2B\right)^2}{2\sqrt{K}(1-\gamma^m)}} \tag{6.36}$$

$$\varepsilon_{\text{app}} = \frac{1+\gamma^m}{1-\gamma^m} \min_{f \in \mathcal{F}_{\mathcal{X}}^B} \|f - Q^\pi\|_d \tag{6.37}$$

$$\varepsilon_{\text{shift}} = \left(B + \frac{1}{1-\gamma}\right) \sqrt{\frac{2\gamma^m}{1-\gamma^m} \sqrt{\|d_m - d\|_{\text{TV}}}} \tag{6.38}$$

$$\varepsilon_{\text{alias}} = \frac{2}{1-\gamma} \left\| \mathbb{E}^\pi \left[\sum_{k=0}^{\infty} \gamma^{km} \left\| \hat{b}_{km} - b_{km} \right\|_{\text{TV}} \middle| Z_0 = \cdot \right] \right\|_d, \tag{6.39}$$

with $d(z,a) = \sum_{s \in \mathcal{S}} d^\pi(s,z)\pi(a|z)$ the sampling distribution, and $d_m(z,a) = \sum_{s \in \mathcal{S}} d_m^\pi(s,z)\pi(a|z)$ the bootstrapping distribution.

The first three terms are identical or analogous to the asymmetric case. The fourth term $\varepsilon_{\text{alias}}$ results from the difference between the fixed point $\widetilde{Q}^\pi$ of the symmetric Bellman operator (6.7) and the true Q-function Q^π.

We note some minor differences with respect to the original result of Cayci et al. [2024] that appear to be typos and minor mistakes in the original proof.[1] We provide the corrected proof in Appendix 6.D.

The results of Theorem 6.3 and Theorem 6.4 can be straightforwardly generalized to any other sampling distribution. However, obtaining bounds in term of $d^\pi \otimes \pi$ is useful for bounding the performance of the actor-critic algorithm.

6.4.3 Finite-Time Bound for the Natural Actor-Critic

Following Cayci et al. [2024], we assume that there exists a concentrability coefficient $\overline{C}_\infty < \infty$ such that $\sup_{0 \leq t < T} \mathbb{E}[C_t] \leq \overline{C}_\infty$ with,

$$C_t = \sup_{s,z,a} \left| \frac{d^{\pi^*}(s,z)\pi^*(a|z)}{d^{\pi_{\theta_t}}(s,z)\pi_{\theta_t}(a|z)} \right|. \tag{6.40}$$

Roughly speaking, this assumption means that all successive policies should visit every agent states and actions visited by the optimal policy with nonzero probability. It motivates the log-linear policy parametrization in equation (6.18) and the initialization to the maximum entropy policy in Algorithm 6.2. We obtain the following finite-time bound for the suboptimality of the policy resulting from Algorithm 6.2.

[1] The authors notably wrongly bound the distance $\|\widehat{Q}_*^\pi - \widetilde{Q}^\pi\|_d$ by ε_{app} at one point, which nevertheless yields a similar result.

Theorem 6.5 (Finite-time bound for asymmetric and symmetric natural actor-critic algorithm). For any agent-state process $\mathcal{M} = (\mathcal{Z}, U)$, we have for Algorithm 6.2 with $\alpha = \frac{1}{\sqrt{K}}$, $\zeta = \frac{B\sqrt{1-\gamma}}{\sqrt{2N}}$, $\eta = \frac{1}{\sqrt{T}}$ and arbitrary $B > 0$,

$$(1-\gamma) \min_{0 \le t < T} \mathbb{E}\left[J(\pi^*) - J(\pi_t)\right] \le \varepsilon_{\text{nac}} + 2\varepsilon_{\text{inf}} + \overline{C}_\infty \left(\varepsilon_{\text{actor}} + 2\varepsilon_{\text{grad}} + 2\sqrt{6} \frac{1}{T} \sum_{t=0}^{T-1} \varepsilon_{\text{critic}}^{\pi_t} \right), \tag{6.41}$$

where the different terms may differ for asymmetric and symmetric critics,

$$\varepsilon_{\text{nac}} = \frac{B^2 + 2\log|\mathcal{A}|}{2\sqrt{T}} \tag{6.42}$$

$$\varepsilon_{\text{actor}} = \sqrt{\frac{(2-\gamma)B}{(1-\gamma)\sqrt{N}}} \tag{6.43}$$

$$\varepsilon_{\text{inf,asym}} = 0 \tag{6.44}$$

$$\varepsilon_{\text{inf,sym}} = \mathbb{E}^{\pi^*}\left[\sum_{k=0}^{\infty} \gamma^k \left\| \hat{b}_k - b_k \right\|_{\text{TV}} \right] \tag{6.45}$$

$$\varepsilon_{\text{grad,asym}} = \sup_{0 \le t < T} \sqrt{\min_w \mathcal{L}_t(w)} \tag{6.46}$$

$$\varepsilon_{\text{grad,sym}} = \sup_{0 \le t < T} \sqrt{\min_w L_t(w)}, \tag{6.47}$$

and $\varepsilon_{\text{critic}}^{\pi_t}$ is given in Theorem 6.3 and Theorem 6.4.

The first term ε_{nac} is the usual natural actor-critic term decreasing in $T^{-1/2}$ [Agarwal et al., 2021]. The second term ε_{inf} is the inference error resulting from use of an agent state in a POMDP [Cayci et al., 2024]. This term is zero for the asymmetric algorithm. The third term $\varepsilon_{\text{actor}}$ is the error resulting from the estimation of the natural policy gradient by stochastic gradient descent. The fourth term $\varepsilon_{\text{grad}}$ is the error resulting from the use of a linear function approximator with features $\nabla_\theta \log \pi_t(a|z)$ for the natural policy gradient. Finally, the fifth term $\frac{1}{T}\sum_{t=0}^{T-1} \varepsilon_{\text{critic}}^{\pi_t}$ is the error arising from the successive critic approximations. Inside of each $\varepsilon_{\text{critic}}^{\pi_t}$ terms, the aliasing term is thus zero for the asymmetric algorithm. The proof, generalizing that of Cayci et al. [2024] to the asymmetric setting, is available in Appendix 6.E.

6.4.4 Discussion

As can be seen from Theorem 6.3 and Theorem 6.4, compared to the symmetric temporal difference learning algorithm, the asymmetric one eliminates a term arising from aliasing in the agent state, in the sense of equation (6.39). In other words, even for an aliased agent-state process, leveraging the state to learn the asymmetric Q-function instead of the symmetric Q-function does not suffer from aliasing, while still providing a valid critic for the policy gradient algorithm. That said, these bounds are given in expectation, and future works may want to study the variance of the error of such Q-function approximations.

From Theorem 6.5, we notice that the inference term (6.45) in the suboptimality bound vanishes in the asymmetric setting. Moreover, the average error $\frac{1}{T}\sum_{t=0}^{T-1}\varepsilon_{\text{critic}}^{\pi_t}$ made in the evaluation of all policies $\pi_0, \dots, \pi_{t-1}$ appears in the finite-time bound that we obtain for the suboptimality of the policy. Thus, the suboptimality bound for the actor also improves in the asymmetric setting by eliminating the aliasing terms with respect to the symmetric setting.

By diving into the proof of Theorem 6.5 at equations (6.232) and (6.233), we understand that the Q-function error impacts the suboptimality bound through the estimation of the natural policy gradient (6.19). Indeed, this error term in the suboptimality bound directly results from the error on the advantage function estimation used in the target of the natural policy gradient estimation loss of equations (6.23) and (6.26). This advantage function estimation is derived from the estimation of the Q-function, such that the error on the latter directly impacts the error on the former, as detailed in equations (6.232) and (6.233). This improvement in the average critic error unfortunately comes at the expense of a different residual error $\varepsilon_{\text{grad}}$ on the natural policy gradient loss. Indeed, as can be seen in equation (6.47), we obtain a residual error $\varepsilon_{\text{grad,asym}}$ using the best approximation of the asymmetric advantage $\mathcal{A}^{\pi_t}(s, z, a)$, instead of a residual error $\varepsilon_{\text{grad,sym}}$ using the best approximation of the symmetric critic $A^{\pi_t}(z, a)$. Since both natural policy gradients are obtained through a linear regression with features $\nabla_\theta \log \pi_t(a|z)$, it is clear than the asymmetric residual error may be higher than the symmetric residual error, even in the tabular case.

We conclude that the effectiveness of asymmetric actor-critic algorithms notably results from a better approximation of the Q-function by eliminating the aliasing bias, which in turn provides a better estimate of the policy gradient.

6.5 Conclusion

In this work, we extended the unbiased asymmetric actor-critic algorithm to agent-state policies. Then, we adapted a finite-time analysis for natural actor-critic to the asymmetric setting. This analysis highlighted that on the contrary to symmetric learning, asymmetric learning is less sensitive to aliasing in the agent state. While this analysis assumed a fixed agent-state process, we argue that it is useful to interpret the causes of effectiveness of asymmetric learning with learnable agent-state processes. Indeed, aliasing can be present in the agent-state process throughout learning, and in particular at initialization. Moreover, it should be noted that this analysis can be straightforwardly generalized to learnable agent-state processes by extending the action space to select future agent states. More formally, we would extend the action space to $\mathcal{A}^+ = \mathcal{A} \times \Delta(\mathcal{Z})$ with $a_t^+ = (a_t, a_t^z)$, the agent state space to $\mathcal{Z}^+ = \mathcal{Z} \times \mathcal{O}$ with $z_t^+ = (z_t, z_t^o)$, and the agent-state process to $U(z_{t+1}^+ | z_t^+, a_t, o_{t+1}) \propto \exp(a_t^{z_{t+1}})\delta_{z_{t+1}^o, o_{t+1}}$. This alternative to backpropagation through time would nevertheless still not reflect the common setting of recurrent actor-critic algorithms. We consider this as a future work that could build on recent advances in finite-time bound for recurrent actor-critic algorithms [Cayci and Eryilmaz, 2024a,b]. Alternatively, generalizing this analysis to nonlinear approximators may include recurrent neural networks, which can be seen as nonlinear approximators with a sliding window as agent state. Our analysis also motivates future

work studying other asymmetric learning approaches that consider representation losses to reduce the aliasing bias [Sinha and Mahajan, 2023, Lambrechts et al., 2022, 2024a].

6.A Agent State Aliasing

In this section, we provide an example of aliased agent state, and discuss the corresponding aliasing bias. For this purpose, we introduce a slightly modified version of the Tiger POMDP [Kaelbling et al., 1998], see Figure 6.1. In this POMDP, there are two doors: one opening on a room with a treasure on the left, and another opening on a room with a tiger on the right. There are four states for this POMDP: being in the treasure room (Treasure), being in the tiger room (Tiger), being in front of the treasure door (Left) or being in front of the tiger door (Right). The rooms are labeled outside (Left or Right), but inside it is completely dark (Dark), such that we do not observe in which room we are. When outside of the rooms, the agent can switch to the other door (Swap) or it can open the door and enter the room (Enter). Once in a room (Treasure or Tiger), the agent stays locked forever, and gets a positive reward (+1) if if it is in the treasure room (Treasure) whatever the action taken (Swap or Enter). We consider the agent state to be simply the last observation (Left, Right, or Dark). Notice that the optimal agent-state policy conditioned on this agent state is also an optimal history-dependent policy. In other words, the current observation is a sufficient statistic for optimal control in this POMDP. We consider a uniform initial distributions over the four states.

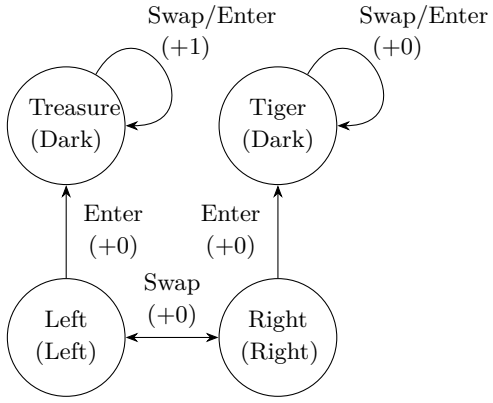


Figure 6.1: Aliased Tiger POMDP.

For a given agent state (Dark), there exist two different underlying states (Treasure or Tiger). We call this phenomenon aliasing. Now, let us consider a simple policy π that always takes the same action (Enter). It is clear that the symmetric value function defined according to equation (6.6) is given by $V^\pi(z = \text{Dark}) = \frac{1}{2(1-\gamma)}$, $V^\pi(z = \text{Left}) = \frac{\gamma}{1-\gamma}$, and $V^\pi(z = \text{Right}) = 0$. However, when considering the unique fixed point of the aliased Bellman operator of equation (6.7) with $m = 1$, we have instead $\widetilde{V}^\pi(z = \text{Dark}) = \frac{1}{2(1-\gamma)}$, $\widetilde{V}^\pi(z = \text{Left}) = \frac{\gamma}{2(1-\gamma)}$, and $\widetilde{V}^\pi(z = \text{Right}) = \frac{\gamma}{2(1-\gamma)}$. We refer to the distance between V^π and $\widetilde{V}^\pi$, or similarly Q^π and $\widetilde{Q}^\pi$, as the aliasing bias. In the analysis of this paper, this distance appears as the weighted ℓ_2-norm $\|Q^\pi - \widetilde{Q}^\pi\|_d$ where $d(s, z, a) = d^\pi(s, z)\pi(a|z)$. In the analysis, we also define the aliasing term $\varepsilon_{\text{alias}}$ as an upper bound on this aliasing bias, see Lemma 6.D.1 for a detailed definition.

6.B Proof of the Natural Policy Gradients

In this section, we prove that the natural policy gradient is the minimizer of analogous asymmetric and symmetric losses.

6.B.1 Proof of the Asymmetric Natural Policy Gradient

In this section, we prove that the natural policy gradient is the minimizer of an asymmetric loss.

Theorem 6.1 (Asymmetric natural policy gradient). For any POMDP $\mathcal{P}$ and any agent-state policy $\pi_\theta \in \Pi_{\mathcal{M}}$, we have,

$$w_*^{\pi_\theta} = (1-\gamma) F_{\pi_\theta}^\dagger \nabla_\theta J(\pi_\theta) \in \operatorname*{arg\,min}_{w \in \mathbb{R}^{d_\psi}} \mathcal{L}(w), \tag{6.21}$$

with,

$$\mathcal{L}(w) = \mathbb{E}^{d^{\pi_\theta}}\Big[\big(\langle \nabla_\theta \log \pi_\theta(A|Z), w\rangle - \mathcal{A}^{\pi_\theta}(S,Z,A)\big)^2\Big]. \tag{6.22}$$

Proof. Let us note that,

$$\nabla_w \mathcal{L}(w) = 2\mathbb{E}^{d^{\pi_\theta}}\left[\nabla_\theta \log \pi_\theta(A|Z)\left(\langle \nabla_\theta \log \pi_\theta(A|Z), w\rangle - \mathcal{A}^{\pi_\theta}(S,Z,A)\right)\right]. \tag{6.48}$$

Therefore, for any $w_*^{\pi_\theta} \in \mathbb{R}^{d_\psi}$ minimizing $\mathcal{L}(w)$, we have $\nabla_w \mathcal{L}(w) = 0$, such that,

$$\begin{aligned}
&\mathbb{E}^{d^{\pi_\theta}}\left[\nabla_\theta \log \pi_\theta(A|Z) \mathcal{A}^{\pi_\theta}(S,Z,A)\right] \\
&\quad = \mathbb{E}^{d^{\pi_\theta}}\left[\nabla_\theta \log \pi_\theta(A|Z) \langle \nabla_\theta \log \pi_\theta(A|Z), w_*^{\pi_\theta}\rangle\right] && (6.49)\\
&\quad = \mathbb{E}^{d^{\pi_\theta}}\left[(\nabla_\theta \log \pi_\theta(A|Z) \otimes \nabla_\theta \log \pi_\theta(A|Z)) w_*^{\pi_\theta}\right] && (6.50)\\
&\quad = \mathbb{E}^{d^{\pi_\theta}}\left[\nabla_\theta \log \pi_\theta(A|Z) \otimes \nabla_\theta \log \pi_\theta(A|Z)\right] w_*^{\pi_\theta} && (6.51)\\
&\quad = F_{\pi_\theta} w_*^{\pi_\theta}. && (6.52)
\end{aligned}$$

which follows from the definition of the Fisher information matrix F_{π_θ} in equation (6.20). Now, let us define the policy $\pi_\theta^+(A|S,Z) = \pi_\theta(A|Z)$, which ignores the state S. From there, we have,

$$\begin{aligned}
F_{\pi_\theta} w_*^{\pi_\theta} &= \mathbb{E}^{d^{\pi_\theta}}\left[\nabla_\theta \log \pi_\theta(A|Z) \mathcal{A}(S,Z,A)\right] && (6.53)\\
&= \mathbb{E}^{d^{\pi_\theta^+}}\left[\nabla_\theta \log \pi_\theta^+(A|S,Z) \mathcal{A}(S,Z,A)\right] && (6.54)\\
&= \mathbb{E}^{d^{\pi_\theta^+}}\left[\nabla_\theta \log \pi_\theta^+(A|S,Z) \left(\mathcal{A}(S,Z,A) + \mathcal{V}(S,Z) - \mathcal{V}(S,Z)\right)\right] && (6.55)\\
&= \mathbb{E}^{d^{\pi_\theta^+}}\left[\nabla_\theta \log \pi_\theta^+(A|S,Z) \mathcal{Q}(S,Z,A)\right] \\
&\qquad - \mathbb{E}^{d^{\pi_\theta^+}}\left[\nabla_\theta \log \pi_\theta^+(A|S,Z) \mathcal{V}(S,Z)\right] && (6.56)\\
&= \mathbb{E}^{d^{\pi_\theta^+}}\left[\nabla_\theta \log \pi_\theta^+(A|S,Z) \mathcal{Q}(S,Z,A)\right] \\
&\qquad - \mathbb{E}^{d^{\pi_\theta^+}}\left[\mathcal{V}(S,Z) \sum_{a \in \mathcal{A}} \pi_\theta^+(a|S,Z) \nabla_\theta \log \pi_\theta^+(a|S,Z)\right] && (6.57)\\
&= \mathbb{E}^{d^{\pi_\theta^+}}\left[\nabla_\theta \log \pi_\theta^+(A|S,Z) \mathcal{Q}(S,Z,A)\right] \\
&\qquad - \mathbb{E}^{d^{\pi_\theta^+}}\left[\mathcal{V}(S,Z) \sum_{a \in \mathcal{A}} \nabla_\theta \pi_\theta^+(a|S,Z)\right] && (6.58)
\end{aligned}$$

$$= \mathbb{E}^{d^{\pi_\theta^+}}\left[\nabla_\theta \log \pi_\theta^+(A|S,Z)\mathcal{Q}(S,Z,A)\right] - \mathbb{E}^{d^{\pi_\theta^+}}\left[\mathcal{V}(S,Z)\nabla_\theta \sum_{a\in\mathcal{A}} \pi_\theta^+(a|S,Z)\right] \tag{6.59}$$

$$= \mathbb{E}^{d^{\pi_\theta^+}}\left[\nabla_\theta \log \pi_\theta^+(A|S,Z)\mathcal{Q}(S,Z,A)\right] - \mathbb{E}^{d^{\pi_\theta^+}}\left[\mathcal{V}(S,Z)\nabla_\theta 1\right] \tag{6.60}$$

$$= \mathbb{E}^{d^{\pi_\theta^+}}\left[\nabla_\theta \log \pi_\theta^+(A|S,Z)\mathcal{Q}(S,Z,A)\right]. \tag{6.61}$$

Using the policy gradient theorem [Sutton et al., 1999] and equation (6.61),

$$F_{\pi_\theta} w_*^{\pi_\theta} = (1-\gamma)\nabla_\theta J(\pi_\theta^+), \tag{6.62}$$

From there, we obtain using the definition of π_θ^+,

$$F_{\pi_\theta} w_*^{\pi_\theta} = (1-\gamma)\nabla_\theta J(\pi_\theta^+) \tag{6.63}$$

$$= (1-\gamma)\nabla_\theta J(\pi_\theta). \tag{6.64}$$

This concludes the proof. □

6.B.2 Proof of the Symmetric Natural Policy Gradient

In this section, we prove that the natural policy gradient is the minimizer of an asymmetric loss.

Theorem 6.2 (Symmetric natural policy gradient). For any POMDP $\mathcal{P}$ and any agent-state policy $\pi_\theta \in \Pi_{\mathcal{M}}$, we have,

$$w_*^{\pi_\theta} = (1-\gamma)F_{\pi_\theta}^\dagger \nabla_\theta J(\pi_\theta) \in \operatorname*{arg\,min}_{w\in\mathbb{R}^{d_\psi}} L(w), \tag{6.24}$$

with,

$$L(w) = \mathbb{E}^{d^{\pi_\theta}}\Big[\left(\langle \nabla_\theta \log \pi_\theta(A|Z), w\rangle - A^{\pi_\theta}(Z,A)\right)^2\Big]. \tag{6.25}$$

Proof. Similarly to the asymmetric setting, for any $w_*^{\pi_\theta}$ minimizing $L(w)$, we have $\nabla_w L(w) = 0$, such that,

$$\mathbb{E}^{d^{\pi_\theta}}[\nabla_\theta \log \pi_\theta(A|Z)A(Z,A)]$$

$$= \mathbb{E}^{d^{\pi_\theta}}[\nabla_\theta \log \pi_\theta(A|Z)\langle \nabla_\theta \log \pi_\theta(A|Z) w_*^{\pi_\theta}\rangle] \tag{6.65}$$

$$= \mathbb{E}^{d^{\pi_\theta}}[(\nabla_\theta \log \pi_\theta(A|Z) \otimes \nabla_\theta \log \pi_\theta(A|Z)) w_*^{\pi_\theta}] \tag{6.66}$$

$$= \mathbb{E}^{d^{\pi_\theta}}[\nabla_\theta \log \pi_\theta(A|Z) \otimes \nabla_\theta \log \pi_\theta(A|Z)]\, w_*^{\pi_\theta} \tag{6.67}$$

$$= F_{\pi_\theta} w_*^{\pi_\theta}, \tag{6.68}$$

which follows from the definition of the Fisher information matrix F_{π_θ} in equation (6.20). From there, we have,

$$F_{\pi_\theta} w_*^{\pi_\theta} = \mathbb{E}^{d^{\pi_\theta}}[\nabla_\theta \log \pi_\theta(A|Z)A(Z,A)] \tag{6.69}$$

$$F_{\pi_\theta} w_*^{\pi_\theta} = \mathbb{E}^{d^{\pi_\theta}}\Big[\nabla_\theta \log \pi_\theta(A|Z)\mathbb{E}^{d^{\pi_\theta}}[\mathcal{A}(S,Z,A)|Z,A]\Big] \tag{6.70}$$

$$F_{\pi_\theta} w_*^{\pi_\theta} = \mathbb{E}^{d^{\pi_\theta}} \left[\mathbb{E}^{d^{\pi_\theta}} [\nabla_\theta \log \pi_\theta(A|Z) \mathcal{A}(S, Z, A) | Z, A] \right] \tag{6.71}$$

$$F_{\pi_\theta} w_*^{\pi_\theta} = \mathbb{E}^{d^{\pi_\theta}} [\nabla_\theta \log \pi_\theta(A|Z) \mathcal{A}(S, Z, A)], \tag{6.72}$$

which follows from the law of total probability. From there, by following the same steps as in the asymmetric case (see Subappendix 6.B.1), we obtain,

$$F_{\pi_\theta} w_*^{\pi_\theta} = (1 - \gamma) \nabla_\theta J(\pi_\theta). \tag{6.73}$$

This concludes the proof. □

6.C Proof of the Finite-Time Bound for the Asymmetric Critic

In this section, we prove Theorem 6.3, that is recalled below.

Theorem 6.3 (Finite-time bound for asymmetric m-step temporal difference learning). For any agent-state policy $\pi \in \Pi_{\mathcal{M}}$, and any $m \in \mathbb{N}$, we have for Algorithm 6.1 with $\alpha = \frac{1}{\sqrt{K}}$ and arbitrary $B > 0$,

$$\sqrt{\mathbb{E}\left[\left\| \mathcal{Q}^\pi - \overline{\mathcal{Q}}^\pi \right\|_d^2 \right]} \leq \varepsilon_{\text{td}} + \varepsilon_{\text{app}} + \varepsilon_{\text{shift}}, \tag{6.29}$$

where the temporal difference learning, function approximation, and distribution shift terms are given by,

$$\varepsilon_{\text{td}} = \sqrt{\frac{4B^2 + \left(\frac{1}{1-\gamma} + 2B \right)^2}{2\sqrt{K}(1 - \gamma^m)}} \tag{6.30}$$

$$\varepsilon_{\text{app}} = \frac{1 + \gamma^m}{1 - \gamma^m} \min_{f \in \mathcal{F}_\phi^B} \| f - \mathcal{Q}^\pi \|_d \tag{6.31}$$

$$\varepsilon_{\text{shift}} = \left(B + \frac{1}{1 - \gamma} \right) \sqrt{\frac{2\gamma^m}{1 - \gamma^m} \sqrt{\| d_m - d \|_{\text{TV}}}}, \tag{6.32}$$

with $d(s, z, a) = d^\pi(s, z)\pi(a|z)$ the sampling distribution, and $d_m(s, z, a) = d_m^\pi(s, z)\pi(a|z)$ the bootstrapping distribution.

Proof. To simplify notation, we drop the dependence on π and β and use $\mathcal{Q}$ as a shorthand for $\mathcal{Q}^\pi$, $\widehat{Q}^*$ as a shorthand for $\widehat{Q}_*^\pi$, $\overline{\mathcal{Q}}$ as a shorthand for $\overline{\mathcal{Q}}^\pi$ and $\widehat{\mathcal{Q}}_k$ as a shorthand for $\widehat{\mathcal{Q}}_{\beta_k}^\pi$, where the subscripts and superscripts remain implicit but are assumed clear from context. When evaluating the Q-functions, we go one step further by using $\mathcal{Q}_{k,i}$ to denote $\mathcal{Q}(S_{k,i}, Z_{k,i}, A_{k,i})$, $\widehat{Q}_{k,i}^*$ to denote $\widehat{Q}^*(Z_{k,i}, A_{k,i})$ or $\widehat{\mathcal{Q}}_{k,i}$ to denote $\widehat{\mathcal{Q}}_k(S_{k,i}, Z_{k,i}, A_{k,i})$, and $\phi_{k,i}$ to denote $\phi(S_{k,i}, Z_{k,i}, A_{k,i})$. In addition, we define d as a shorthand for $d^\pi \otimes \pi$, such that $d(s, z, a) = d^\pi(s, z)\pi(a|z)$, and d_m as a shorthand for $d_m^\pi \otimes \pi$, such that $d_m(s, z, a) = d_m^\pi(s, z)\pi(a|z)$.

First, let us define Δ_k as,

$$\Delta_k = \sqrt{\mathbb{E}\left[\left\| \mathcal{Q} - \widehat{\mathcal{Q}}_k \right\|_d^2 \right]} = \sqrt{\mathbb{E}\left[\| \mathcal{Q}(\cdot) - \langle \beta_k, \phi(\cdot) \rangle \|_d^2 \right]}. \tag{6.74}$$

Using the linearity of $\overline{\mathcal{Q}}$ in $\beta_1, \ldots, \beta_{K-1}$, the triangle inequality, the subadditivity of the square root, and Jensen's inequality, we have,

$$\sqrt{\mathbb{E}\left[\left\|\mathcal{Q}-\overline{\mathcal{Q}}\right\|_d^2\right]} = \sqrt{\mathbb{E}\left[\left\|\mathcal{Q}(\cdot)-\left\langle\frac{1}{K}\sum_{k=0}^{K-1}\beta_k,\phi(\cdot)\right\rangle\right\|_d^2\right]} \tag{6.75}$$

$$= \sqrt{\mathbb{E}\left[\left\|\frac{1}{K}\sum_{k=0}^{K-1}\left(\mathcal{Q}(\cdot)-\langle\beta_k,\phi(\cdot)\rangle\right)\right\|_d^2\right]} \tag{6.76}$$

$$= \sqrt{\mathbb{E}\left[\left\|\sum_{k=0}^{K-1}\frac{1}{K}\left(\mathcal{Q}(\cdot)-\langle\beta_k,\phi(\cdot)\rangle\right)\right\|_d^2\right]} \tag{6.77}$$

$$\leq \sqrt{\mathbb{E}\left[\sum_{k=0}^{K-1}\frac{1}{K^2}\left\|\mathcal{Q}(\cdot)-\langle\beta_k,\phi(\cdot)\rangle\right\|_d^2\right]} \tag{6.78}$$

$$= \sqrt{\frac{1}{K^2}\sum_{k=0}^{K-1}\mathbb{E}\left[\left\|\mathcal{Q}(\cdot)-\langle\beta_k,\phi(\cdot)\rangle\right\|_d^2\right]} \tag{6.79}$$

$$= \frac{1}{K}\sqrt{\sum_{k=0}^{K-1}\Delta_k^2} \tag{6.80}$$

$$\leq \frac{1}{K}\sum_{k=0}^{K-1}\sqrt{\Delta_k^2} \tag{6.81}$$

$$= \frac{1}{K}\sum_{k=0}^{K-1}\Delta_k \tag{6.82}$$

$$= \frac{1}{K}\sum_{k=0}^{K-1}(\Delta_k - l) + l \tag{6.83}$$

$$\leq \sqrt{\left(\frac{1}{K}\sum_{k=0}^{K-1}(\Delta_k - l)\right)^2} + l \tag{6.84}$$

$$\leq \sqrt{\frac{1}{K}\sum_{k=0}^{K-1}(\Delta_k - l)^2} + l, \tag{6.85}$$

where l is arbitrary.

Now, we consider the Lyapounov function $\mathcal{L}(\beta) = \|\beta_* - \beta\|_2^2$ in order to find a bound on $\frac{1}{K}\sum_{k=0}^{K-1}(\Delta_k - l)^2$. Since $\beta_* \in \mathcal{B}_2(0,B)$, with $\mathcal{B}_2(0,B)$ a convex subset of $\mathbb{R}^{d_\phi}$, and the projection $\Gamma_{\mathcal{C}}$ is non-expansive for closed and convex $\mathcal{C}$, we have for all $k \geq 0$,

$$\mathcal{L}(\beta_{k+1}) = \|\beta_* - \beta_{k+1}\|_2^2 \tag{6.86}$$

$$\leq \left\|\beta_* - \beta_{k+1}^-\right\|_2^2 \tag{6.87}$$

$$= \|\beta_* - (\beta_k + \alpha g_k)\|_2^2 \tag{6.88}$$

$$= \|(\beta_* - \beta_k) - \alpha g_k\|_2^2 \tag{6.89}$$
$$= \langle (\beta_* - \beta_k) - \alpha g_k, (\beta_* - \beta_k) - \alpha g_k \rangle \tag{6.90}$$
$$= \langle \beta_* - \beta_k, \beta_* - \beta_k \rangle - 2\alpha \langle \beta_* - \beta_k, g_k \rangle + \alpha^2 \langle g_k, g_k \rangle \tag{6.91}$$
$$= \mathcal{L}(\beta_k) - 2\alpha \langle \beta_* - \beta_k, g_k \rangle + \alpha^2 \|g_k\|_2^2 \tag{6.92}$$
$$= \mathcal{L}(\beta_k) + 2\alpha \langle \beta_k - \beta_*, g_k \rangle + \alpha^2 \|g_k\|_2^2 . \tag{6.93}$$

Let us consider the Lyapounov drift $\mathbb{E}[\mathcal{L}(\beta_{k+1}) - \mathcal{L}(\beta_k)]$, and exploit the fact that environments samples used to compute g_k are independent and identically distributed. Formally, we define $\mathfrak{G}_k = \sigma(S_{i,j}, Z_{i,j}, A_{i,j}, i \leq k, j \leq m)$ and $\mathfrak{F}_k = \sigma(S_{k,0}, Z_{k,0}, A_{k,0})$, where $\sigma(X_i : i \in \mathcal{I})$ denotes the σ-algebra generated by a collection $\{X_i \colon i \in \mathcal{I}\}$ of random variables. We can write, using to the law of total expectation,

$$\begin{aligned}
&\mathbb{E}[\mathcal{L}(\beta_{k+1}) - \mathcal{L}(\beta_k)] \\
&\quad = \mathbb{E}\left[\mathbb{E}[\mathcal{L}(\beta_{k+1}) - \mathcal{L}(\beta_k) | \mathfrak{G}_{k-1}]\right] && (6.94) \\
&\quad \leq 2\alpha \mathbb{E}\left[\mathbb{E}[\langle \beta_k - \beta_*, g_k \rangle | \mathfrak{G}_{k-1}]\right] + \alpha^2 \mathbb{E}\left[\mathbb{E}\left[\|g_k\|_2^2 \middle| \mathfrak{G}_{k-1}\right]\right]. && (6.95)
\end{aligned}$$

Let us focus on the first term of equation (6.95) with $\mathbb{E}[\langle g_k, \beta_k - \beta_* \rangle | \mathfrak{G}_{k-1}]$. First, since $\nabla_\beta \widehat{\mathcal{Q}}_{k,0} = \phi_{k,0}$, the semi-gradient g_k is given by (see equation (6.13)),

$$g_k = \left(\sum_{t=0}^{m-1} \gamma^t R_{k,t} + \gamma^m \widehat{\mathcal{Q}}_{k,m} - \widehat{\mathcal{Q}}_{k,0} \right) \phi_{k,0}. \tag{6.96}$$

By conditioning on the sigma-fields $\mathfrak{G}_{k-1}$ and $\mathfrak{F}_k$, we have,

$$\begin{aligned}
&\mathbb{E}[\langle \beta_k - \beta_*, g_k \rangle | \mathfrak{F}_k, \mathfrak{G}_{k-1}] \\
&\quad = \left(\mathbb{E}\left[\sum_{t=0}^{m-1} \gamma^t R_{k,t} + \gamma^m \widehat{\mathcal{Q}}_{k,m} \middle| \mathfrak{F}_k, \mathfrak{G}_{k-1} \right] - \widehat{\mathcal{Q}}_{k,0} \right) \langle \beta_k - \beta_*, \phi_{k,0} \rangle && (6.97) \\
&\quad = \left(\mathbb{E}\left[\sum_{t=0}^{m-1} \gamma^t R_{k,t} + \gamma^m \widehat{\mathcal{Q}}_{k,m} \middle| \mathfrak{F}_k, \mathfrak{G}_{k-1} \right] - \widehat{\mathcal{Q}}_{k,0} \right) \left(\widehat{\mathcal{Q}}_{k,0} - \widehat{\mathcal{Q}}^*_{k,0} \right). && (6.98)
\end{aligned}$$

Note that according to the Bellman operator (6.5) we have,

$$\mathbb{E}\left[\sum_{t=0}^{m-1} \gamma^t R_{k,t} \middle| \mathfrak{F}_k, \mathfrak{G}_{k-1} \right] = \mathcal{Q}_{k,0} - \gamma^m \mathbb{E}[\mathcal{Q}_{k,m} | \mathfrak{F}_k, \mathfrak{G}_{k-1}]. \tag{6.99}$$

By substituting equation (6.99) in equation (6.98), we obtain,

$$\begin{aligned}
&\mathbb{E}[\langle \beta_k - \beta_*, g_k \rangle | \mathfrak{F}_k, \mathfrak{G}_{k-1}] \\
&\quad = \left(\widehat{\mathcal{Q}}_{k,0} - \widehat{\mathcal{Q}}^*_{k,0} \right) \left(\mathbb{E}\left[\sum_{t=0}^{m-1} \gamma^t R_{k,t} \middle| \mathfrak{F}_k, \mathfrak{G}_{k-1} \right] \right. \\
&\qquad\qquad \left. + \gamma^m \mathbb{E}\left[\widehat{\mathcal{Q}}_{k,m} \middle| \mathfrak{F}_k, \mathfrak{G}_{k-1} \right] - \widehat{\mathcal{Q}}_{k,0} \right) && (6.100)
\end{aligned}$$

$$= \left(\widehat{\mathcal{Q}}_{k,0} - \widehat{\mathcal{Q}}^*_{k,0}\right) \Bigg(\mathcal{Q}_{k,0} - \gamma^m \mathbb{E}\left[\mathcal{Q}_{k,m} | \mathfrak{F}_k, \mathfrak{G}_{k-1}\right] + \gamma^m \mathbb{E}\left[\widehat{\mathcal{Q}}_{k,m} \Big| \mathfrak{F}_k, \mathfrak{G}_{k-1}\right] - \widehat{\mathcal{Q}}_{k,0} \Bigg) \tag{6.101}$$

$$= \left((\widehat{\mathcal{Q}}_{k,0} - \mathcal{Q}_{k,0}) + (\mathcal{Q}_{k,0} - \widehat{\mathcal{Q}}^*_{k,0})\right) \Bigg((\mathcal{Q}_{k,0} - \widehat{\mathcal{Q}}_{k,0}) - \gamma^m \mathbb{E}\left[\mathcal{Q}_{k,m} - \widehat{\mathcal{Q}}_{k,m} \Big| \mathfrak{F}_k, \mathfrak{G}_{k-1}\right] \Bigg) \tag{6.102}$$

$$\begin{aligned} &= -(\mathcal{Q}_{k,0} - \widehat{\mathcal{Q}}_{k,0})^2 + (\mathcal{Q}_{k,0} - \widehat{\mathcal{Q}}_{k,0})(\mathcal{Q}_{k,0} - \widehat{\mathcal{Q}}^*_{k,0}) \\ &\quad + \gamma^m \mathbb{E}\left[\widehat{\mathcal{Q}}_{k,m} - \mathcal{Q}_{k,m} \Big| \mathfrak{F}_k, \mathfrak{G}_{k-1}\right] (\widehat{\mathcal{Q}}_{k,0} - \mathcal{Q}_{k,0}) \\ &\quad + \gamma^m \mathbb{E}\left[\widehat{\mathcal{Q}}_{k,m} - \mathcal{Q}_{k,m} \Big| \mathfrak{F}_k, \mathfrak{G}_{k-1}\right] (\mathcal{Q}_{k,0} - \widehat{\mathcal{Q}}^*_{k,0}). \end{aligned} \tag{6.103}$$

Let us now take the expectation of (6.103) over $\mathfrak{F}_k$ given $\mathfrak{G}_{k-1}$, for each term separately,

- For the first term, we have,

$$\mathbb{E}\left[-(\mathcal{Q}_{k,0} - \widehat{\mathcal{Q}}_{k,0})^2 \Big| \mathfrak{G}_{k-1}\right] = -\left\|\mathcal{Q} - \widehat{\mathcal{Q}}_k\right\|_d^2. \tag{6.104}$$

- For the second term, we have, using the Cauchy-Schwarz inequality,

$$\begin{aligned} &\mathbb{E}\left[(\mathcal{Q}_{k,0} - \widehat{\mathcal{Q}}_{k,0})(\mathcal{Q}_{k,0} - \widehat{\mathcal{Q}}^*_{k,0}) \Big| \mathfrak{G}_{k-1}\right] \\ &\qquad = \left\|(\mathcal{Q} - \widehat{\mathcal{Q}}_k)(\mathcal{Q} - \widehat{\mathcal{Q}}^*)\right\|_d \qquad (6.105) \\ &\qquad \leq \left\|\mathcal{Q} - \widehat{\mathcal{Q}}_k\right\|_d \left\|\mathcal{Q} - \widehat{\mathcal{Q}}^*\right\|_d. \qquad (6.106) \end{aligned}$$

Before proceeding to the third and fourth terms, let us notice that,

$$\begin{aligned} &\mathbb{E}\left[\widehat{\mathcal{Q}}_{k,m} - \mathcal{Q}_{k,m} \Big| \mathfrak{G}_{k-1}\right] \\ &\quad = \sum_{s,z,a} d_m(s,z,a) \left(\widehat{\mathcal{Q}}_k(s,z,a) - \mathcal{Q}(s,z,a)\right) \qquad (6.107) \\ &\quad = \sum_{s,z,a} (d(s,z,a) + d_m(s,z,a) - d(s,z,a)) \left(\widehat{\mathcal{Q}}_k(s,z,a) - \mathcal{Q}(s,z,a)\right). \qquad (6.108) \end{aligned}$$

Remembering that $\sup_{s,z,a} \widehat{\mathcal{Q}}_k(s,z,a) \leq B$ and $\sup_{s,z,a} \mathcal{Q}(s,z,a) \leq \frac{1}{1-\gamma}$, we have,

$$\begin{aligned} &\mathbb{E}\left[\left(\widehat{\mathcal{Q}}_{k,m} - \mathcal{Q}_{k,m}\right)^2 \Big| \mathfrak{G}_{k-1}\right] \\ &\quad = \sum_{s,z,a} (d(s,z,a) + d_m(s,z,a) - d(s,z,a)) \left(\widehat{\mathcal{Q}}_k(s,z,a) - \mathcal{Q}(s,z,a)\right)^2 \end{aligned}$$

$$= \left\| \widehat{\mathcal{Q}}_k - \mathcal{Q} \right\|_d^2 + \sum_{s,z,a} (d_m(s,z,a) - d(s,z,a)) \left(\widehat{\mathcal{Q}}_k(s,z,a) - \mathcal{Q}(s,z,a) \right)^2$$

$$\leq \left\| \widehat{\mathcal{Q}}_k - \mathcal{Q} \right\|_d^2 + \|d_m - d\|_{\mathrm{TV}} \sup_{s,z,a} \left(\widehat{\mathcal{Q}}_k(s,z,a) - \mathcal{Q}(s,z,a) \right)^2$$

$$\leq \left\| \widehat{\mathcal{Q}}_k - \mathcal{Q} \right\|_d^2 + \|d_m - d\|_{\mathrm{TV}} \left(B + \frac{1}{1-\gamma} \right)^2, \tag{6.109}$$

where $\left(B + \frac{1}{1-\gamma} \right)$ is an upper bound on $\sup_{s,z,a} \left| \widehat{\mathcal{Q}}_k(s,z,a) - \mathcal{Q}(s,z,a) \right|$. Now, using Jensen's inequality and the subadditivity of the square root, we have,

$$\mathbb{E}\left[\widehat{\mathcal{Q}}_{k,m} - \mathcal{Q}_{k,m} \middle| \mathfrak{G}_{k-1} \right]$$

$$\leq \mathbb{E}\left[\sqrt{(\widehat{\mathcal{Q}}_{k,m} - \mathcal{Q}_{k,m})^2} \middle| \mathfrak{G}_{k-1} \right] \tag{6.110}$$

$$\leq \sqrt{\mathbb{E}\left[\left(\widehat{\mathcal{Q}}_{k,m} - \mathcal{Q}_{k,m} \right)^2 \middle| \mathfrak{G}_{k-1} \right]} \tag{6.111}$$

$$\leq \left\| \widehat{\mathcal{Q}}_k - \mathcal{Q} \right\|_d + \left(B + \frac{1}{1-\gamma} \right) \sqrt{\|d_m - d\|_{\mathrm{TV}}}. \tag{6.112}$$

With this, we proceed to the third and fourth terms (without the multiplier γ^m) and show the following.

- For the third term, we have by upper bounding $|\widehat{\mathcal{Q}}_{k,0} - \mathcal{Q}_{k,0}|$ by $B + \frac{1}{1-\gamma}$,

$$\mathbb{E}\left[(\widehat{\mathcal{Q}}_{k,m} - \mathcal{Q}_{k,m})(\widehat{\mathcal{Q}}_{k,0} - \mathcal{Q}_{k,0}) \middle| \mathfrak{G}_{k-1} \right]$$

$$\leq \left\| \widehat{\mathcal{Q}}_k - \mathcal{Q} \right\|_d^2 + \left(B + \frac{1}{1-\gamma} \right)^2 \sqrt{\|d_m - d\|_{\mathrm{TV}}}. \tag{6.113}$$

- For the fourth term, we have by upper bounding $|\mathcal{Q}_{k,0} - \widehat{\mathcal{Q}}^*_{k,0}|$ by $\frac{1}{1-\gamma} + B$,

$$\mathbb{E}\left[(\widehat{\mathcal{Q}}_{k,m} - \mathcal{Q}_{k,m})(\mathcal{Q}_{k,0} - \widehat{\mathcal{Q}}^*_{k,0}) \middle| \mathfrak{G}_{k-1} \right]$$

$$\leq \left\| \widehat{\mathcal{Q}}_k - \mathcal{Q} \right\|_d \left\| \mathcal{Q} - \widehat{\mathcal{Q}}^* \right\|_d + \left(B + \frac{1}{1-\gamma} \right)^2 \sqrt{\|d_m - d\|_{\mathrm{TV}}}. \tag{6.114}$$

By taking expectation over $\mathfrak{G}_{k-1}$ of the four terms and using the previous upper bounds, we obtain,

$$\mathbb{E}\left[\langle \beta_k - \beta_*, g_k \rangle \right] = \mathbb{E}\left[\mathbb{E}\left[\langle \beta_k - \beta_*, g_k \rangle | \mathfrak{G}_{k-1} \right] \right] \tag{6.115}$$

$$\leq -(1-\gamma^m)\mathbb{E}\left[\left\| \widehat{\mathcal{Q}}_k - \mathcal{Q} \right\|_d^2 \right] + (1+\gamma^m)\mathbb{E}\left[\left\| \widehat{\mathcal{Q}}_k - \mathcal{Q} \right\|_d \right] \left\| \widehat{\mathcal{Q}}^* - \mathcal{Q} \right\|_d$$
$$+ 2\gamma^m \left(B + \frac{1}{1-\gamma} \right)^2 \sqrt{\|d_m - d\|_{\mathrm{TV}}} \tag{6.116}$$

$$= -(1-\gamma^m)\Delta_k^2 + (1+\gamma^m)\Delta_k \left\| \widehat{\mathcal{Q}}^* - \mathcal{Q} \right\|_d$$
$$+ 2\gamma^m \left(B + \frac{1}{1-\gamma} \right)^2 \sqrt{\|d_m - d\|_{\mathrm{TV}}}. \tag{6.117}$$

Let us now focus on the second term of equation (6.95) with $\mathbb{E}\left[\|g_k\|_2^2 \middle| \mathfrak{G}_{k-1}\right]$. Since $\sup_{s,z,a} \|\phi(s,z,a)\|_2 \leq 1$ and $\|\beta_k\|_2 \leq B$ for all $k \geq 0$, and $r_{k,i} \leq 1$ for all $k \geq 0$ and for all $i < m-1$, the norm of the gradient (6.96) is bounded as follows,

$$\sup_{k\geq 0} \|g_k\|_2 \leq \frac{1-\gamma^m}{1-\gamma} + (1+\gamma^m)B \leq \frac{1}{1-\gamma} + 2B. \tag{6.118}$$

We obtain, for the second term of equation (6.95),

$$\mathbb{E}\left[\|g_k\|_2^2\right] = \mathbb{E}\left[\mathbb{E}\left[\|g_k\|_2^2 \middle| \mathfrak{G}_{k-1}\right]\right] \tag{6.119}$$

$$\leq \left(\frac{1}{1-\gamma} + 2B\right)^2. \tag{6.120}$$

By substituting equations (6.117) and (6.120) into the Lyapounov drift of equation (6.95), we obtain,

$$\begin{aligned}\mathbb{E}\left[\mathcal{L}(\beta_{k+1}) - \mathcal{L}(\beta_k)\right] \leq & -2\alpha(1-\gamma^m)\Delta_k^2 + 2\alpha(1+\gamma^m)\Delta_k \left\|\widehat{\mathcal{Q}}^* - \mathcal{Q}\right\|_d \\ & + \alpha^2 \left(\frac{1}{1-\gamma} + 2B\right)^2 \\ & + 4\alpha\gamma^m \left(B + \frac{1}{1-\gamma}\right)^2 \sqrt{\|d_m - d\|_{\mathrm{TV}}}. \end{aligned} \tag{6.121}$$

By setting $l = \frac{1+\gamma^m}{2(1-\gamma^m)} \min_{f\in\mathcal{F}_\phi^B} \|f - \mathcal{Q}\|_d$, we can write,

$$\begin{aligned}&\mathbb{E}\left[\mathcal{L}(\beta_{k+1}) - \mathcal{L}(\beta_k)\right] \\ &\quad \leq -2\alpha(1-\gamma^m)\left(\Delta_k^2 - 2l\Delta_k\right) + \alpha^2\left(\frac{1}{1-\gamma} + 2B\right)^2 \\ &\qquad + 4\alpha\gamma^m\left(B + \frac{1}{1-\gamma}\right)^2 \sqrt{\|d_m - d\|_{\mathrm{TV}}} \qquad (6.122) \\ &\quad = -2\alpha(1-\gamma^m)\left(\Delta_k^2 - 2l\Delta_k + l^2\right) + 2\alpha(1-\gamma^m)l^2 + \alpha^2\left(\frac{1}{1-\gamma} + 2B\right)^2 \\ &\qquad + 4\alpha\gamma^m\left(B + \frac{1}{1-\gamma}\right)^2 \sqrt{\|d_m - d\|_{\mathrm{TV}}} \qquad (6.123) \\ &\quad = -2\alpha(1-\gamma^m)\left(\Delta_k - l\right)^2 + 2\alpha(1-\gamma^m)l^2 + \alpha^2\left(\frac{1}{1-\gamma} + 2B\right)^2 \\ &\qquad + 4\alpha\gamma^m\left(B + \frac{1}{1-\gamma}\right)^2 \sqrt{\|d_m - d\|_{\mathrm{TV}}}. \qquad (6.124)\end{aligned}$$

By summing all Lyapounov drifts $\sum_{k=0}^{K-1} \mathbb{E}\left[\mathcal{L}(\beta_{k+1}) - \mathcal{L}(\beta_k)\right]$, we get,

$$\begin{aligned}&\mathbb{E}\left[\mathcal{L}(\beta_K) - \mathcal{L}(\beta_0)\right] \\ &\quad \leq -2\alpha(1-\gamma^m)\sum_{k=0}^{K-1}\left(\Delta_k - l\right)^2 + 2\alpha K(1-\gamma^m)l^2 + \alpha^2 K\left(\frac{1}{1-\gamma} + 2B\right)^2\end{aligned}$$

$$+ 4\alpha K\gamma^m \left(B + \frac{1}{1-\gamma}\right)^2 \sqrt{\|d_m - d\|_{\mathrm{TV}}}. \tag{6.125}$$

By rearranging and dividing by $2\alpha K(1-\gamma^m)$, we obtain after neglecting $\mathcal{L}(\beta_K) > 0$,

$$\begin{aligned}
\frac{1}{K}\sum_{k=0}^{K-1}(\Delta_k - l)^2 &\leq \frac{\mathbb{E}\left[\mathcal{L}(\beta_0) - \mathcal{L}(\beta_K)\right]}{2\alpha K(1-\gamma^m)} + l^2 + \frac{\alpha}{2(1-\gamma^m)}\left(\frac{1}{1-\gamma} + 2B\right)^2 \\
&\quad + \frac{2\gamma^m}{1-\gamma^m}\left(B + \frac{1}{1-\gamma}\right)^2 \sqrt{\|d_m - d\|_{\mathrm{TV}}} \qquad (6.126)\\
&\leq \frac{\|\beta_0 - \beta_*\|_2^2}{2\alpha K(1-\gamma^m)} + l^2 + \frac{\alpha}{2(1-\gamma^m)}\left(\frac{1}{1-\gamma} + 2B\right)^2 \\
&\quad + \frac{2\gamma^m}{1-\gamma^m}\left(B + \frac{1}{1-\gamma}\right)^2 \sqrt{\|d_m - d\|_{\mathrm{TV}}}. \qquad (6.127)
\end{aligned}$$

The bound obtained through this Lyapounov drift summation can be used to further develop equation (6.85), using the subadditivity of the square root,

$$\begin{aligned}
\sqrt{\mathbb{E}\left[\left\|\mathcal{Q} - \overline{\mathcal{Q}}\right\|_d^2\right]} &\leq \sqrt{\frac{1}{K}\sum_{k=0}^{K-1}(\Delta_k - l)^2} + l \qquad (6.128)\\
&\leq \frac{\|\beta_0 - \beta_*\|_2}{\sqrt{2\alpha K(1-\gamma^m)}} + 2l + \sqrt{\frac{\alpha}{2(1-\gamma^m)}}\left(\frac{1}{1-\gamma} + 2B\right) \\
&\quad + \left(B + \frac{1}{1-\gamma}\right)\sqrt{\frac{2\gamma^m}{1-\gamma^m}\sqrt{\|d_m - d\|_{\mathrm{TV}}}} \qquad (6.129)\\
&= \frac{\|\beta_0 - \beta_*\|_2}{\sqrt{2\alpha K(1-\gamma^m)}} + \frac{1+\gamma^m}{1-\gamma^m}\min_{f\in\mathcal{F}_\phi^B}\|f - \mathcal{Q}\|_d \\
&\quad + \sqrt{\frac{\alpha}{2(1-\gamma^m)}}\left(\frac{1}{1-\gamma} + 2B\right) \\
&\quad + \left(B + \frac{1}{1-\gamma}\right)\sqrt{\frac{2\gamma^m}{1-\gamma^m}\sqrt{\|d_m - d\|_{\mathrm{TV}}}}. \qquad (6.130)
\end{aligned}$$

By setting $\alpha = \frac{1}{\sqrt{K}}$ and upper bounding $\|\beta_0 - \beta_*\|$ by $2B$, we get,

$$\begin{aligned}
\sqrt{\mathbb{E}\left[\left\|\mathcal{Q} - \overline{\mathcal{Q}}\right\|_d^2\right]} &\leq \frac{2B}{\sqrt{2\sqrt{K}(1-\gamma^m)}} + \frac{1+\gamma^m}{1-\gamma^m}\min_{f\in\mathcal{F}_\phi^B}\|f - \mathcal{Q}\|_d \\
&\quad + \frac{1}{\sqrt{2\sqrt{K}(1-\gamma^m)}}\left(\frac{1}{1-\gamma} + 2B\right) \\
&\quad + \left(B + \frac{1}{1-\gamma}\right)\sqrt{\frac{2\gamma^m}{1-\gamma^m}\sqrt{\|d_m - d\|_{\mathrm{TV}}}} \qquad (6.131)\\
&= \sqrt{\frac{4B^2 + \left(\frac{1}{1-\gamma} + 2B\right)^2}{2\sqrt{K}(1-\gamma^m)}} + \frac{1+\gamma^m}{1-\gamma^m}\min_{f\in\mathcal{F}_\phi^B}\|f - \mathcal{Q}\|_d
\end{aligned}$$

$$+\left(B+\frac{1}{1-\gamma}\right)\sqrt{\frac{2\gamma^m}{1-\gamma^m}\sqrt{\|d_m-d\|_{\mathrm{TV}}}}. \tag{6.132}$$

This concludes the proof. □

6.D Proof of the Finite-Time Bound for the Symmetric Critic

Let us first find an upper bound on the distance $\left\|Q^\pi-\widetilde{Q}^\pi\right\|_d^2$ between the Q-function Q^π and the fixed point $\widetilde{Q}^\pi$.

Lemma 6.D.1 (Upper bound on the aliasing bias [Cayci et al., 2024]). For any agent-state policy $\pi\in\Pi_{\mathcal{M}}$, and any $m\in\mathbb{N}$, we have,

$$\left\|Q^\pi-\widetilde{Q}^\pi\right\|_d\leq\frac{1-\gamma^m}{1-\gamma}\left\|\mathbb{E}^\pi\left[\sum_{k=0}^{\infty}\gamma^{km}\left\|\hat{b}_{km}-b_{km}\right\|_{\mathrm{TV}}\middle|Z_0=\cdot\right]\right\|_d. \tag{6.133}$$

Proof. The proof is similar to the one of Cayci et al. [2024]. Let us first define the expected m-step return,

$$\bar{r}_m(s,z,a)=\mathbb{E}^\pi\left[\sum_{k=0}^{m-1}\gamma^k R_k\middle|S_0=s,Z_0=s,A_0=a\right]. \tag{6.134}$$

Using the expected m-step return and the definition of the belief b in equation (6.33) and approximate belief $\hat{b}$ in equation (6.34), it can be noted that,

$$Q^\pi(z,a)=\mathbb{E}^\pi\left[\sum_{k=0}^{\infty}\gamma^{km}\sum_{s\in\mathcal{S}}b_{km}(s|H_{km})\bar{r}_m(s,Z_{km},A_{km})\middle|Z_0=z,A_0=a\right] \tag{6.135}$$

$$\widetilde{Q}^\pi(z,a)=\mathbb{E}^\pi\left[\sum_{k=0}^{\infty}\gamma^{km}\sum_{s\in\mathcal{S}}\hat{b}_{km}(s|Z_{km})\bar{r}_m(s,Z_{km},A_{km})\middle|Z_0=z,A_0=a\right]. \tag{6.136}$$

Indeed, bootstrapping at time step m based on the agent state only is equivalent to considering the distribution of future states to be $\hat{b}_m(\cdot|Z_m)$ instead of $b_m(\cdot|H_m)$. As a consequence, we have,

$$\begin{aligned}
&\left|Q^\pi(z,a)-\widetilde{Q}^\pi(z,a)\right| \\
&\quad=\mathbb{E}^\pi\left[\sum_{k=0}^{\infty}\gamma^{km}\sum_{s\in\mathcal{S}}\left(b_{km}(s|H_{km})-\hat{b}_{km}(s|Z_{km})\right)\right. \\
&\qquad\left.\bar{r}_m(s,Z_{km},A_{km})\middle|Z_0=z,A_0=a\right] \\
&\quad\leq\mathbb{E}^\pi\left[\sum_{k=0}^{\infty}\gamma^{km}\sup_{s\in\mathcal{S}}\left|b_{km}(s|H_{km})-\hat{b}_{km}(s|Z_{km})\right|\right.
\end{aligned} \tag{6.137}$$

$$\sup_{s\in\mathcal{S}}|\bar{r}_m(s,Z_{km},A_{km})|\,\Big|\,Z_0=z,A_0=a\Big] \tag{6.138}$$

$$\leq \mathbb{E}^\pi\Bigg[\sum_{k=0}^{\infty}\gamma^{km}\sup_{s\in\mathcal{S}}\Big|b_{km}(s|H_{km})-\hat{b}_{km}(s|Z_{km})\Big|\,\frac{1-\gamma^m}{1-\gamma}$$

$$\Big|\,Z_0=z,A_0=a\Bigg] \tag{6.139}$$

$$=\frac{1-\gamma^m}{1-\gamma}\mathbb{E}^\pi\Bigg[\sum_{k=0}^{\infty}\gamma^{km}\sup_{s\in\mathcal{S}}\Big|b_{km}(s|H_{km})-\hat{b}_{km}(s|Z_{km})\Big|$$

$$\Big|\,Z_0=z,A_0=a\Bigg] \tag{6.140}$$

$$\leq\frac{1-\gamma^m}{1-\gamma}\mathbb{E}^\pi\left[\sum_{k=0}^{\infty}\gamma^{km}\left\|b_{km}(\cdot|H_{km})-\hat{b}_{km}(\cdot|Z_{km})\right\|_{\mathrm{TV}}\middle|Z_0=z,A_0=a\right] \tag{6.141}$$

$$\leq\frac{1-\gamma^m}{1-\gamma}\mathbb{E}^\pi\left[\sum_{k=0}^{\infty}\gamma^{km}\left\|b_{km}-\hat{b}_{km}\right\|_{\mathrm{TV}}\middle|Z_0=z,A_0=a\right], \tag{6.142}$$

where we use b_{km} and $\hat{b}_{km}$ to denote the random variables $b_{km}(\cdot|H_{km})$ and $\hat{b}_{km}(\cdot|Z_{km})$, respectively. It illustrates that the aliasing bias can be bounded proportionally to the distance between the true belief and the approximate belief at the bootstrapping time steps. Then, we obtain,

$$\left\|Q^\pi-\widetilde{Q}^\pi\right\|_d\leq\frac{1-\gamma^m}{1-\gamma}\left\|\mathbb{E}^\pi\left[\sum_{k=0}^{\infty}\gamma^{km}\left\|\hat{b}_{km}-b_{km}\right\|_{\mathrm{TV}}\middle|Z_0=\cdot\right]\right\|_d. \tag{6.143}$$

This concludes the proof. □

Using Lemma 6.D.1, we can prove Theorem 6.4, that is recalled below. Note that some notations used in Appendix 6.C will be reused with another meaning.

Theorem 6.4 (Finite-time bound for symmetric m-step temporal difference learning [Cayci et al., 2024]). For any agent-state policy $\pi\in\Pi_{\mathcal{M}}$, and any $m\in\mathbb{N}$, we have for Algorithm 6.1 with $\alpha=\frac{1}{\sqrt{K}}$, and arbitrary $B>0$,

$$\sqrt{\mathbb{E}\left[\left\|Q^\pi-\overline{Q}^\pi\right\|_d^2\right]}\leq\varepsilon_{\mathrm{td}}+\varepsilon_{\mathrm{app}}+\varepsilon_{\mathrm{shift}}+\varepsilon_{\mathrm{alias}}, \tag{6.35}$$

where the temporal difference learning, function approximation, distribution shift, and aliasing terms are given by,

$$\varepsilon_{\mathrm{td}}=\sqrt{\frac{4B^2+\left(\frac{1}{1-\gamma}+2B\right)^2}{2\sqrt{K}(1-\gamma^m)}} \tag{6.36}$$

$$\varepsilon_{\mathrm{app}}=\frac{1+\gamma^m}{1-\gamma^m}\min_{f\in\mathcal{F}_{\mathcal{X}}^B}\|f-Q^\pi\|_d \tag{6.37}$$

$$\varepsilon_{\text{shift}} = \left(B + \frac{1}{1-\gamma}\right)\sqrt{\frac{2\gamma^m}{1-\gamma^m}\sqrt{\|d_m - d\|_{\text{TV}}}} \tag{6.38}$$

$$\varepsilon_{\text{alias}} = \frac{2}{1-\gamma}\left\|\mathbb{E}^{\pi}\left[\sum_{k=0}^{\infty}\gamma^{km}\left\|\hat{b}_{km} - b_{km}\right\|_{\text{TV}}\middle| Z_0 = \cdot\right]\right\|_d, \tag{6.39}$$

with $d(z,a) = \sum_{s\in\mathcal{S}} d^{\pi}(s,z)\pi(a|z)$ the sampling distribution, and $d_m(z,a) = \sum_{s\in\mathcal{S}} d_m^{\pi}(s,z)\pi(a|z)$ the bootstrapping distribution.

Proof. To ease notation as for the proof of Theorem 6.3 in Appendix 6.C, we use Q as a shorthand for Q^{π}, $\widehat{Q}^*$ as a shorthand for $\widehat{Q}_*^{\pi}$, $\widetilde{Q}$ as a shorthand for $\widetilde{Q}^{\pi}$, $\overline{Q}$ as a shorthand for $\overline{Q}^{\pi}$ and $\widehat{Q}_k$ as a shorthand for $\widehat{Q}_{\beta_k}^{\pi}$, where the subscripts and superscripts remain implicit but are assumed clear from context. When evaluating the Q-functions, we go one step further by using $Q_{k,i}$ to denote $Q(Z_{k,i}, A_{k,i})$, $\widehat{Q}_{k,i}^*$ to denote $\widehat{Q}^*(Z_{k,i}, A_{k,i})$, $\widetilde{Q}_{k,i}$ to denote $\widetilde{Q}(Z_{k,i}, A_{k,i})$ and $\widehat{Q}_{k,i}$ to denote $\widehat{Q}_k(Z_{k,i}, A_{k,i})$, and $\chi_{k,i}$ to denote $\chi(Z_{k,i}, A_{k,i})$. In addition, we define d as a shorthand for $d^{\pi} \otimes \pi$, such that $d(z,a) = d^{\pi}(z)\pi(a|z)$, and d_m as a shorthand for $d_m^{\pi} \otimes \pi$, such that $d_m(z,a) = d_m^{\pi}(z)\pi(a|z)$. Using the triangle inequality and the subadditivity of the square root, we have,

$$\sqrt{\mathbb{E}\left[\|Q - \overline{Q}\|_d^2\right]} \leq \sqrt{\mathbb{E}\left[\left\|Q - \widetilde{Q}\right\|_d^2\right] + \mathbb{E}\left[\left\|\widetilde{Q} - \overline{Q}\right\|_d^2\right]} \tag{6.144}$$

$$\leq \sqrt{\mathbb{E}\left[\left\|Q - \widetilde{Q}\right\|_d^2\right]} + \sqrt{\mathbb{E}\left[\left\|\widetilde{Q} - \overline{Q}\right\|_d^2\right]} \tag{6.145}$$

$$\leq \left\|Q - \widetilde{Q}\right\|_d + \sqrt{\mathbb{E}\left[\left\|\widetilde{Q} - \overline{Q}\right\|_d^2\right]}. \tag{6.146}$$

We can bound the second term in equation (6.146) using similar steps as in the proof for the asymmetric finite-time bound (see Appendix 6.C). We obtain,

$$\sqrt{\mathbb{E}\left[\left\|\widetilde{Q} - \overline{Q}\right\|_d^2\right]} \leq \sqrt{\frac{1}{K}\sum_{k=0}^{K-1}(\Delta_k - l)^2 + l}, \tag{6.147}$$

where l is arbitrary, and Δ_k is defined as,

$$\Delta_k = \sqrt{\mathbb{E}\left[\left\|\widetilde{Q} - \widehat{Q}_k\right\|_d^2\right]} = \sqrt{\mathbb{E}\left[\left\|\widetilde{Q}(\cdot) - \langle\beta_k, \chi(\cdot)\rangle\right\|_d^2\right]}. \tag{6.148}$$

Similarly to the asymmetric case (see Appendix 6.C), we consider the Lyapounov function $\mathcal{L}(\beta) = \|\beta_* - \beta\|_2^2$ in order to find a bound on $\frac{1}{K}\sum_{k=0}^{K-1}(\Delta_k - l)^2$. We define $\mathfrak{G}_k = \sigma(Z_{i,j}, A_{i,j}, i \leq k, j \leq m)$ and $\mathfrak{F}_k = \sigma(Z_{k,0}, A_{k,0})$. As in the asymmetric case (see Appendix 6.C), we obtain, using to the law of total expectation,

$$\mathbb{E}\left[\mathcal{L}(\beta_{k+1}) - \mathcal{L}(\beta_k)\right] \leq 2\alpha\mathbb{E}\left[\mathbb{E}\left[\langle\beta_k - \beta_*, g_k\rangle|\mathfrak{G}_{k-1}\right]\right] + \alpha^2\mathbb{E}\left[\mathbb{E}\left[\|g_k\|_2^2\middle|\mathfrak{G}_{k-1}\right]\right]. \tag{6.149}$$

Let us focus on the first term of equation (6.149) with $\mathbb{E}\left[\langle \beta_k - \beta_*, g_k \rangle | \mathfrak{G}_{k-1}\right]$. By conditioning on the sigma-fields $\mathfrak{G}_{k-1}$ and $\mathfrak{F}_k$, we have,

$$\begin{aligned}
&\mathbb{E}\left[\langle \beta_k - \beta_*, g_k \rangle | \mathfrak{F}_k, \mathfrak{G}_{k-1}\right] \\
&\quad = \left(\mathbb{E}\left[\sum_{t=0}^{m-1} \gamma^t R_{k,t} + \gamma^m \widehat{Q}_{k,m} \middle| \mathfrak{F}_k, \mathfrak{G}_{k-1} \right] - \widehat{Q}_{k,0} \right) \left(\widehat{Q}_{k,0} - \widehat{Q}^*_{k,0} \right).
\end{aligned} \tag{6.150}$$

Note that, according to the Bellman operator (6.7), we have,

$$\mathbb{E}\left[\sum_{t=0}^{m-1} \gamma^t R_{k,t} \middle| \mathfrak{F}_k, \mathfrak{G}_{k-1} \right] = \widetilde{Q}_{k,0} - \gamma^m \mathbb{E}\left[\widetilde{Q}_{k,m} \middle| \mathfrak{F}_k, \mathfrak{G}_{k-1} \right]. \tag{6.151}$$

It differs from the asymmetric case (see Appendix 6.C) in that we do not necessarily have $Q = \widetilde{Q}$ here. By substituting equation (6.151) in equation (6.150), we obtain,

$$\begin{aligned}
&\mathbb{E}\left[\langle \beta_k - \beta_*, g_k \rangle | \mathfrak{F}_k, \mathfrak{G}_{k-1}\right] \\
&\quad = \left(\widehat{Q}_{k,0} - \widehat{Q}^*_{k,0} \right) \left(\mathbb{E}\left[\sum_{t=0}^{m-1} \gamma^t R_{k,t} \middle| \mathfrak{F}_k, \mathfrak{G}_{k-1} \right] \right. \\
&\qquad\qquad \left. + \gamma^m \mathbb{E}\left[\widehat{Q}_{k,m} \middle| \mathfrak{F}_k, \mathfrak{G}_{k-1} \right] - \widehat{Q}_{k,0} \right) \qquad (6.152) \\
&\quad = \left(\widehat{Q}_{k,0} - \widehat{Q}^*_{k,0} \right) \left(\widetilde{Q}_{k,0} - \gamma^m \mathbb{E}\left[\widetilde{Q}_{k,m} \middle| \mathfrak{F}_k, \mathfrak{G}_{k-1} \right] \right. \\
&\qquad\qquad \left. + \gamma^m \mathbb{E}\left[\widehat{Q}_{k,m} \middle| \mathfrak{F}_k, \mathfrak{G}_{k-1} \right] - \widehat{Q}_{k,0} \right) \qquad (6.153) \\
&\quad = \left((\widehat{Q}_{k,0} - \widetilde{Q}_{k,0}) + (\widetilde{Q}_{k,0} - \widehat{Q}^*_{k,0}) \right) \left((\widetilde{Q}_{k,0} - \widehat{Q}_{k,0}) \right. \\
&\qquad\qquad \left. - \gamma^m \mathbb{E}\left[\widetilde{Q}_{k,m} - \widehat{Q}_{k,m} \middle| \mathfrak{F}_k, \mathfrak{G}_{k-1} \right] \right) \qquad (6.154) \\
&\quad = -(\widetilde{Q}_{k,0} - \widehat{Q}_{k,0})^2 + (\widetilde{Q}_{k,0} - \widehat{Q}_{k,0})(\widetilde{Q}_{k,0} - \widehat{Q}^*_{k,0}) \\
&\qquad + \gamma^m \mathbb{E}\left[\widehat{Q}_{k,m} - \widetilde{Q}_{k,m} \middle| \mathfrak{F}_k, \mathfrak{G}_{k-1} \right] (\widehat{Q}_{k,0} - \widetilde{Q}_{k,0}) \\
&\qquad + \gamma^m \mathbb{E}\left[\widehat{Q}_{k,m} - \widetilde{Q}_{k,m} \middle| \mathfrak{F}_k, \mathfrak{G}_{k-1} \right] (\widetilde{Q}_{k,0} - \widehat{Q}^*_{k,0}). \qquad (6.155)
\end{aligned}$$

We now follow the same technique as in the asymmetric case (see Appendix 6.C) for each of the four terms. By taking the expectation over $\mathfrak{F}_k$, we get the following.

- For the first term, we have,

$$\mathbb{E}\left[-(\widetilde{Q}_{k,0} - \widehat{Q}_{k,0})^2 \middle| \mathfrak{G}_{k-1} \right] = -\left\| \widetilde{Q} - \widehat{Q}_k \right\|_d^2. \tag{6.156}$$

- For the second term, we have,

$$\mathbb{E}\left[(\widetilde{Q}_{k,0}-\widehat{Q}_{k,0})(\widetilde{Q}_{k,0}-\widehat{Q}^*_{k,0})\middle|\mathfrak{G}_{k-1}\right] \leq \left\|\widetilde{Q}-\widehat{Q}_k\right\|_d \left\|\widetilde{Q}-\widehat{Q}^*\right\|_d. \tag{6.157}$$

- For the third term, we have,

$$\mathbb{E}\left[(\widehat{Q}_{k,m}-\widetilde{Q}_{k,m})(\widehat{Q}_{k,0}-\widetilde{Q}_{k,0})\middle|\mathfrak{G}_{k-1}\right] \leq \left\|\widehat{Q}_k-\widetilde{Q}\right\|_d^2 + \left(B+\frac{1}{1-\gamma}\right)^2\sqrt{\|d_m-d\|_{\mathrm{TV}}}. \tag{6.158}$$

- For the fourth term, we have,

$$\mathbb{E}\left[(\widehat{Q}_{k,m}-\widetilde{Q}_{k,m})(\widetilde{Q}_{k,0}-\widehat{Q}^*_{k,0})\middle|\mathfrak{G}_{k-1}\right] \leq \left\|\widehat{Q}_k-\widetilde{Q}\right\|_d \left\|\widetilde{Q}-\widehat{Q}^*\right\|_d + \left(B+\frac{1}{1-\gamma}\right)^2\sqrt{\|d_m-d\|_{\mathrm{TV}}}. \tag{6.159}$$

By taking expectation over $\mathfrak{G}_{k-1}$ of the four terms and using the previous upper bounds, we obtain,

$$\mathbb{E}\left[\langle\beta_k-\beta_*,g_k\rangle\right] \leq -(1-\gamma^m)\Delta_k^2 + (1+\gamma^m)\Delta_k\left\|\widehat{Q}^*-\widetilde{Q}\right\|_d + 2\gamma^m\left(B+\frac{1}{1-\gamma}\right)^2\sqrt{\|d_m-d\|_{\mathrm{TV}}}. \tag{6.160}$$

The second term in equation (6.149) is treated similarly to the asymmetric case (see Appendix 6.C), which yields,

$$\mathbb{E}\left[\|g_k\|_2^2\right] \leq \left(\frac{1}{1-\gamma}+2B\right)^2. \tag{6.161}$$

By substituting equations (6.160) and (6.161) into the Lyapounov drift of equation (6.149), we obtain,

$$\mathbb{E}\left[\mathcal{L}(\beta_{k+1})-\mathcal{L}(\beta_k)\right] \leq -2\alpha(1-\gamma^m)\Delta_k^2 + 2\alpha(1+\gamma^m)\Delta_k\left\|\widehat{Q}^*-\widetilde{Q}\right\|_d + \alpha^2\left(\frac{1}{1-\gamma}+2B\right)^2 + 4\alpha\gamma^m\left(B+\frac{1}{1-\gamma}\right)^2\sqrt{\|d_m-d\|_{\mathrm{TV}}}. \tag{6.162}$$

We can upper bound $\left\|\widehat{Q}^*-\widetilde{Q}\right\|_d$ as follows,

$$\left\|\widehat{Q}^*-\widetilde{Q}\right\|_d \leq \left\|\widehat{Q}^*-Q\right\|_d + \left\|Q-\widetilde{Q}\right\|_d. \tag{6.163}$$

By setting $l = \frac{1+\gamma^m}{2(1-\gamma^m)} \left(\left\| \widehat{Q}^* - Q \right\|_d + \left\| Q - \widetilde{Q} \right\|_d \right)$, we can write, following a similar strategy as in the asymmetric case (see Appendix 6.C),

$$\begin{aligned}
&\mathbb{E}\left[\mathcal{L}(\beta_{k+1}) - \mathcal{L}(\beta_k)\right] \\
&\quad \leq -2\alpha(1-\gamma^m)\left(\Delta_k - l\right)^2 + 2\alpha(1-\gamma^m)l^2 + \alpha^2\left(\frac{1}{1-\gamma} + 2B\right)^2 \\
&\qquad + 4\alpha\gamma^m\left(B + \frac{1}{1-\gamma}\right)^2 \sqrt{\|d_m - d\|_{\mathrm{TV}}}. \qquad (6.164)
\end{aligned}$$

By summing all drifts, rearranging, and dividing by $2\alpha K(1-\gamma^m)$, we obtain after neglecting $\mathcal{L}(\beta_K) > 0$,

$$\begin{aligned}
\frac{1}{K}\sum_{k=0}^{K-1}(\Delta_k - l)^2 \leq \frac{\|\beta_0 - \beta_*\|_2^2}{2\alpha K(1-\gamma^m)} + l^2 + \frac{\alpha}{2(1-\gamma^m)}\left(\frac{1}{1-\gamma} + 2B\right)^2 \\
+ \frac{2\gamma^m}{1-\gamma^m}\left(B + \frac{1}{1-\gamma}\right)^2 \sqrt{\|d_m - d\|_{\mathrm{TV}}}. \qquad (6.165)
\end{aligned}$$

The bound obtained through this Lyapounov drift summation can be used to further develop equation (6.147), using the subadditivity of the square root,

$$\begin{aligned}
\sqrt{\mathbb{E}\left[\left\|\widetilde{Q} - \overline{Q}\right\|_d^2\right]} &\leq \sqrt{\frac{1}{K}\sum_{k=0}^{K-1}\left(\Delta_k - l\right)^2} + l \qquad (6.166) \\
&\leq \frac{\|\beta_0 - \beta_*\|_2}{\sqrt{2\alpha K(1-\gamma^m)}} + 2l + \sqrt{\frac{\alpha}{2(1-\gamma^m)}}\left(\frac{1}{1-\gamma} + 2B\right) \\
&\quad + \left(B + \frac{1}{1-\gamma}\right)\sqrt{\frac{2\gamma^m}{1-\gamma^m}\sqrt{\|d_m - d\|_{\mathrm{TV}}}} \qquad (6.167) \\
&= \frac{\|\beta_0 - \beta_*\|_2}{\sqrt{2\alpha K(1-\gamma^m)}} + 2l + \sqrt{\frac{\alpha}{2(1-\gamma^m)}}\left(\frac{1}{1-\gamma} + 2B\right) \\
&\quad + \left(B + \frac{1}{1-\gamma}\right)\sqrt{\frac{2\gamma^m}{1-\gamma^m}\sqrt{\|d_m - d\|_{\mathrm{TV}}}}. \qquad (6.168)
\end{aligned}$$

Plugging equation (6.168) into equation (6.146), and substituting back l, we finally have,

$$\begin{aligned}
\sqrt{\mathbb{E}\left[\left\|Q - \overline{Q}\right\|_d^2\right]} &\leq \frac{\|\beta_0 - \beta_*\|_2}{\sqrt{2\alpha K(1-\gamma^m)}} + \frac{1+\gamma^m}{1-\gamma^m}\left(\left\|\widehat{Q}^* - Q\right\|_d + \left\|Q - \widetilde{Q}\right\|_d\right) \\
&\quad + \sqrt{\frac{\alpha}{2(1-\gamma^m)}}\left(\frac{1}{1-\gamma} + 2B\right) \\
&\quad + \left(B + \frac{1}{1-\gamma}\right)\sqrt{\frac{2\gamma^m}{1-\gamma^m}\sqrt{\|d_m - d\|_{\mathrm{TV}}}} + \left\|Q - \widetilde{Q}\right\|_d \qquad (6.169) \\
&\leq \frac{\|\beta_0 - \beta_*\|_2}{\sqrt{2\alpha K(1-\gamma^m)}} + \frac{1+\gamma^m}{1-\gamma^m}\left\|\widehat{Q}^* - Q\right\|_d
\end{aligned}$$

$$+\sqrt{\frac{\alpha}{2(1-\gamma^m)}}\left(\frac{1}{1-\gamma}+2B\right)+\frac{2}{1-\gamma^m}\left\|Q-\widetilde{Q}\right\|_d$$
$$+\left(B+\frac{1}{1-\gamma}\right)\sqrt{\frac{2\gamma^m}{1-\gamma^m}\sqrt{\|d_m-d\|_{\mathrm{TV}}}} \tag{6.170}$$

Using Lemma 6.D.1, we finally obtain,

$$\begin{aligned}
&\sqrt{\mathbb{E}\left[\left\|Q-\overline{Q}\right\|_d^2\right]} \\
&\quad\leq \frac{\|\beta_0-\beta_*\|_2}{\sqrt{2\alpha K(1-\gamma^m)}}+\frac{1+\gamma^m}{1-\gamma^m}\left\|\widehat{Q}^*-Q\right\|_d \\
&\qquad+\sqrt{\frac{\alpha}{2(1-\gamma^m)}}\left(\frac{1}{1-\gamma}+2B\right) \\
&\qquad+\left(B+\frac{1}{1-\gamma}\right)\sqrt{\frac{2\gamma^m}{1-\gamma^m}\sqrt{\|d_m-d\|_{\mathrm{TV}}}} \\
&\qquad+\left(\frac{2}{1-\gamma^m}\right)\frac{1-\gamma^m}{1-\gamma}\left\|\mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^{km}\left\|\hat{b}_{km}-b_{km}\right\|_{\mathrm{TV}}\middle| Z_0=\cdot\right]\right\|_d
\end{aligned} \tag{6.171}$$

$$\begin{aligned}
&\leq \frac{\|\beta_0-\beta_*\|_2}{\sqrt{2\alpha K(1-\gamma^m)}}+\frac{1+\gamma^m}{1-\gamma^m}\min_{f\in\mathcal{F}_\phi^B}\|f-\mathcal{Q}\|_d \\
&\qquad+\sqrt{\frac{\alpha}{2(1-\gamma^m)}}\left(\frac{1}{1-\gamma}+2B\right) \\
&\qquad+\left(B+\frac{1}{1-\gamma}\right)\sqrt{\frac{2\gamma^m}{1-\gamma^m}\sqrt{\|d_m-d\|_{\mathrm{TV}}}} \\
&\qquad+\frac{2}{1-\gamma}\left\|\mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^{km}\left\|\hat{b}_{km}-b_{km}\right\|_{\mathrm{TV}}\middle| Z_0=\cdot\right]\right\|_d.
\end{aligned} \tag{6.172}$$

By setting $\alpha=\frac{1}{\sqrt{K}}$ and upper bounding $\|\beta_0-\beta_*\|$ by $2B$, we get,

$$\begin{aligned}
\sqrt{\mathbb{E}\left[\left\|Q-\overline{Q}\right\|_d^2\right]} \leq &\sqrt{\frac{4B^2+\left(\frac{1}{1-\gamma}+2B\right)^2}{2\sqrt{K}(1-\gamma^m)}}+\frac{1+\gamma^m}{1-\gamma^m}\min_{f\in\mathcal{F}_\phi^B}\|f-\mathcal{Q}\|_d \\
&+\left(B+\frac{1}{1-\gamma}\right)\sqrt{\frac{2\gamma^m}{1-\gamma^m}\sqrt{\|d_m-d\|_{\mathrm{TV}}}} \\
&+\frac{2}{1-\gamma}\left\|\mathbb{E}\left[\sum_{k=0}^{\infty}\gamma^{km}\left\|\hat{b}_{km}-b_{km}\right\|_{\mathrm{TV}}\middle| Z_0=\cdot\right]\right\|_d.
\end{aligned} \tag{6.173}$$

This concludes the proof. □

6.E Proof of the Finite-Time Bound for the Natural Actor-Critic

Let us first give the performance difference lemma for POMDP proved by Cayci et al. [2024]. Note that this proof is completely agnostic about the critic used

to compute $\pi_1, \pi_2 \in \Pi_{\mathcal{M}}$ and is thus applicable both to the asymmetric setting and the symmetric setting.

Lemma 6.E.1 (Performance difference [Cayci et al., 2024]). For any two agent-state polices $\pi_1, \pi_2 \in \Pi_{\mathcal{M}}$,

$$V^{\pi_2}(z_0) - V^{\pi_1}(z_0) \leq \frac{1}{1-\gamma}\mathbb{E}^{d^{\pi_2}}\left[A^{\pi_1}(Z,A)|Z_0 = z_0\right] + \frac{2}{1-\gamma}\varepsilon_{\text{inf}}^{\pi_2}(z_0), \quad (6.174)$$

where,

$$\varepsilon_{\text{inf}}^{\pi_2}(z_0) = \mathbb{E}^{\pi_2}\left[\sum_{k=0}^{\infty}\gamma^k \left\|\hat{b}_k - b_k\right\|_{\text{TV}} \middle| Z_0 = z_0\right]. \quad (6.175)$$

Proof. The proof is similar to the one of Cayci et al. [2024]. First, let us decompose the performance difference in the following terms,

$$\begin{aligned}
&V^{\pi_2}(z_0) - V^{\pi_1}(z_0) \\
&\quad = \mathbb{E}^{\pi_2}\left[\sum_{t=0}^{\infty}\gamma^t R_t \middle| Z_0 = z_0\right] - V^{\pi_1}(z_0) && (6.176)\\
&\quad = \mathbb{E}^{\pi_2}\left[\sum_{t=0}^{\infty}\gamma^t \left(R_t - V^{\pi_1}(Z_t) + V^{\pi_1}(Z_t)\right) \middle| Z_0 = z_0\right] - V^{\pi_1}(z_0) && (6.177)\\
&\quad = \mathbb{E}^{\pi_2}\left[\sum_{t=0}^{\infty}\gamma^t \left(R_t - V^{\pi_1}(Z_t) + \gamma V^{\pi_1}(Z_{t+1})\right) \middle| Z_0 = z_0\right] && (6.178)\\
&\quad = \mathbb{E}^{\pi_2}\left[\sum_{t=0}^{\infty}\gamma^t \left(R_t + \gamma\mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1}) - V^{\pi_1}(Z_t)\right) \middle| Z_0 = z_0\right] \\
&\qquad + \mathbb{E}^{\pi_2}\left[\sum_{t=0}^{\infty}\gamma^t \left(\gamma V^{\pi_1}(Z_{t+1}) - \gamma\mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1})\right) \middle| Z_0 = z_0\right] && (6.179)\\
&\quad = \mathbb{E}^{\pi_2}\left[\sum_{t=0}^{\infty}\gamma^t \left(R_t + \gamma\mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1}) - V^{\pi_1}(Z_t)\right) \middle| Z_0 = z_0\right] \\
&\qquad + \mathbb{E}^{\pi_2}\left[\sum_{t=0}^{\infty}\gamma^{t+1} \left(V^{\pi_1}(Z_{t+1}) - \mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1})\right) \middle| Z_0 = z_0\right]. && (6.180)
\end{aligned}$$

Let us focus on bounding the first term in equation (6.180). We have, for any $T > 0$,

$$\left|\sum_{t=0}^{T}\gamma^t \left(R_t + \gamma\mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1}) - V^{\pi_1}(Z_t)\right)\right| \leq \frac{2}{(1-\gamma)^2} < \infty. \quad (6.181)$$

By Lebesgue's dominated convergence, we have,

$$\begin{aligned}
&\mathbb{E}^{\pi_2}\left[\sum_{t=0}^{\infty}\gamma^t \left(R_t + \gamma\mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1}) - V^{\pi_1}(Z_t)\right) \middle| Z_0 = z_0\right] \\
&\quad = \sum_{t=0}^{\infty}\gamma^t \mathbb{E}^{\pi_2}\left[R_t + \gamma\mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1}) - V^{\pi_1}(Z_t)|Z_0 = z_0\right]. && (6.182)
\end{aligned}$$

Then, by the law of total expectation, we have at any time step $t \geq 0$,

$$\begin{aligned}
&\mathbb{E}^{\pi_2}\left[R_t + \gamma \mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1}) - V^{\pi_1}(Z_t) | Z_0 = z_0\right] \\
&\quad = \mathbb{E}\left[\mathbb{E}^{\pi_2}\left[R_t + \gamma \mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1}) | H_t, Z_t\right] - V^{\pi_1}(Z_t) \middle| Z_0 = z_0\right]. \quad (6.183)
\end{aligned}$$

And, we have,

$$\begin{aligned}
&\mathbb{E}^{\pi_2}\left[R_t + \gamma \mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1}) | H_t = h_t, Z_t = z_t\right] \\
&\quad = \sum_{s_t, a_t} b_t(s_t|h_t) \pi_2(a_t|z_t) \mathcal{Q}^{\pi_1}(s_t, z_t, a_t) \quad (6.184) \\
&\quad = \sum_{a_t} \pi_2(a_t|z_t) Q^{\pi_1}(z_t, a_t) + \sum_{s_t, a_t} b_t(s_t|h_t) \pi_2(a_t|z_t) \mathcal{Q}^{\pi_1}(s_t, z_t, a_t) \\
&\qquad - \sum_{a_t} \pi_2(a_t|z_t) Q^{\pi_1}(z_t, a_t) \quad (6.185) \\
&\quad = \sum_{a_t} \pi_2(a_t|z_t) Q^{\pi_1}(z_t, a_t) + \sum_{s_t, a_t} b_t(s_t|h_t) \pi_2(a_t|z_t) \mathcal{Q}^{\pi_1}(s_t, z_t, a_t) \\
&\qquad - \sum_{s_t, a_t} \hat{b}_t(s_t|z_t) \pi_2(a_t|z_t) \mathcal{Q}^{\pi_1}(s_t, z_t, a_t) \quad (6.186) \\
&\quad = \sum_{a_t} \pi_2(a_t|z_t) Q^{\pi_1}(z_t, a_t) \\
&\qquad + \sum_{s_t, a_t} \left(b_t(s_t|h_t) - \hat{b}_t(s_t|z_t)\right) \pi_2(a_t|z_t) \mathcal{Q}^{\pi_1}(s_t, z_t, a_t). \quad (6.187)
\end{aligned}$$

By noting that $\sup_{s,z} \left|\sum_a \pi_2(a|z) \mathcal{Q}^{\pi_1}(s, z, a)\right| \leq \sup_{s,z,a} \left|\mathcal{Q}^{\pi_1}(s, z, a)\right| \leq \frac{1}{1-\gamma}$, we obtain,

$$\begin{aligned}
&\mathbb{E}^{\pi_2}\left[R_t + \gamma \mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1}) | H_t = h_t, Z_t = z_t\right] \\
&\quad \leq \sum_{a_t} \pi_2(a_t|z_t) Q^{\pi_1}(z_t, a_t) + \frac{1}{1-\gamma} \left\| b_t(\cdot|h_t) - \hat{b}_t(\cdot|z_t) \right\|_{\mathrm{TV}}. \quad (6.188)
\end{aligned}$$

Finally, the expectation at time $t \geq 0$ can be written as,

$$\begin{aligned}
&\mathbb{E}^{\pi_2}\left[R_t + \gamma \mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1}) - V^{\pi_1}(Z_t) | Z_0 = z_0\right] \\
&\quad = \mathbb{E}\left[\mathbb{E}^{\pi_2}\left[R_t + \gamma \mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1}) | H_t, Z_t\right] - V^{\pi_1}(Z_t) \middle| Z_0 = z_0\right] \quad (6.189) \\
&\quad \leq \mathbb{E}^{\pi_2}\left[Q^{\pi_1}(Z_t, A_t) + \frac{1}{1-\gamma} \left\| b_t(\cdot|H_t) - \hat{b}_t(\cdot|Z_t) \right\|_{\mathrm{TV}} - V^{\pi_1}(Z_t) \;\middle|\; Z_0 = z_0\right] \quad (6.190) \\
&\quad = \mathbb{E}^{\pi_2}\left[A^{\pi_1}(Z_t, A_t) - \frac{1}{1-\gamma} \left\| b_t(\cdot|H_t) - \hat{b}_t(\cdot|Z_t) \right\|_{\mathrm{TV}} \middle| Z_0 = z_0\right] \quad (6.191)
\end{aligned}$$

Now, by using Lebesgue's dominated theorem in the reverse direction, we have,

$$\begin{aligned}
&\mathbb{E}^{\pi_2}\left[\sum_{t=0}^{\infty} \gamma^t \left(R_t + \gamma \mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1}) - V^{\pi_1}(Z_t)\right) \middle| Z_0 = z_0\right] \\
&\quad \leq \mathbb{E}^{\pi_2}\left[\sum_{t=0}^{\infty} \gamma^t A^{\pi_1}(Z_t, A_t) \middle| Z_0 = z_0\right]
\end{aligned}$$

$$+ \frac{1}{1-\gamma}\mathbb{E}^{\pi_2}\left[\sum_{t=0}^{\infty}\gamma^t \left\|\hat{b}_t - b_t\right\|_{\mathrm{TV}} \middle| Z_0 = z_0\right] \tag{6.192}$$

$$= \mathbb{E}^{\pi_2}\left[\sum_{t=0}^{\infty}\gamma^t A^{\pi_1}(Z_t, A_t) \middle| Z_0 = z_0\right] + \frac{1}{1-\gamma}\varepsilon_{\mathrm{inf}}^{\pi_2}(z_0) \tag{6.193}$$

Now, let us focus on bounding the second term in equation (6.180). We have, for any $T > 0$,

$$\left|\sum_{t=0}^{T}\gamma^{t+1}\left(V^{\pi_1}(Z_{t+1}) - \mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1})\right)\right| \leq \frac{2}{(1-\gamma)^2} < \infty. \tag{6.194}$$

Using Lebesgue dominated convergence theorem, we can write,

$$\mathbb{E}^{\pi_2}\left[\sum_{t=0}^{\infty}\gamma^{t+1}\left(V^{\pi_1}(Z_{t+1}) - \mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1})\right) \middle| Z_0 = z_0\right]$$
$$= \sum_{t=0}^{\infty}\gamma^{t+1}\mathbb{E}^{\pi_2}\left[V^{\pi_1}(Z_{t+1}) - \mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1}) | Z_0 = z_0\right]. \tag{6.195}$$

By the law of total expectation, we have at any time step $t \geq 0$,

$$\mathbb{E}^{\pi_2}\left[V^{\pi_1}(Z_{t+1}) - \mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1}) | Z_0 = z_0\right]$$
$$= \mathbb{E}\left[V^{\pi_1}(Z_{t+1}) - \mathbb{E}^{\pi_2}\left[\mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1}) | H_{t+1}, Z_{t+1}\right] \middle| Z_0 = z_0\right]. \tag{6.196}$$

And, we have,

$$\mathbb{E}^{\pi_2}\left[\mathcal{V}^{\pi_1}(S_{t+1}, z_{t+1}) | H_{t+1} = h_{t+1}, Z_{t+1} = z_{t+1},\right]$$
$$= \sum_{s_{t+1}} b_{t+1}(s_{t+1}|h_{t+1})\mathcal{V}^{\pi_1}(s_{t+1}, z_{t+1}) \tag{6.197}$$

$$= V^{\pi_1}(z_{t+1}) + \sum_{s_{t+1}} b_{t+1}(s_{t+1}|h_{t+1})\mathcal{V}^{\pi_1}(s_{t+1}, z_{t+1}) - V^{\pi_1}(z_{t+1}) \tag{6.198}$$

$$= V^{\pi_1}(z_{t+1}) + \sum_{s_{t+1}} b_{t+1}(s_{t+1}|h_{t+1})\mathcal{V}^{\pi_1}(s_{t+1}, z_{t+1})$$
$$- \sum_{s_{t+1}} \hat{b}_{t+1}(s_{t+1}|z_{t+1})\mathcal{V}^{\pi_1}(s_{t+1}, z_{t+1}) \tag{6.199}$$

$$= V^{\pi_1}(z_{t+1}) + \sum_{s_{t+1}} \left(b_{t+1}(s_{t+1}|h_{t+1}) - \hat{b}_{t+1}(s_{t+1}|z_{t+1})\right)\mathcal{V}^{\pi_1}(s_{t+1}, z_{t+1}). \tag{6.200}$$

From there, by noting that $\sup_{s,z}|\mathcal{V}^{\pi_1}(s,z)| \leq \frac{1}{1-\gamma}$, we obtain,

$$\mathbb{E}^{\pi_2}\left[\mathcal{V}^{\pi_1}(S_{t+1}, z_{t+1}) | H_{t+1} = h_{t+1}, Z_{t+1} = z_{t+1},\right]$$
$$\geq V^{\pi_1}(z_{t+1}) - \frac{1}{1-\gamma}\left\|b_{t+1}(\cdot|h_{t+1}) - \hat{b}_{t+1}(\cdot|z_{t+1})\right\|_{\mathrm{TV}}. \tag{6.201}$$

Finally, the expectation at time $t \geq 0$ can be written as,

$$\mathbb{E}^{\pi_2}\left[V^{\pi_1}(Z_{t+1}) - \mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1}) | Z_0 = z_0\right]$$

$$= \mathbb{E}\left[V^{\pi_1}(Z_{t+1}) - \mathbb{E}^{\pi_2}\left[\mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1})|H_{t+1}, Z_{t+1}\right]\middle| Z_0 = z_0\right] \tag{6.202}$$

$$\leq \mathbb{E}\Bigg[V^{\pi_1}(Z_{t+1}) - V^{\pi_1}(Z_{t+1}) + \frac{1}{1-\gamma}\left\|b_{t+1}(\cdot|H_{t+1}) - \hat{b}_{t+1}(\cdot|Z_{t+1})\right\|_{\mathrm{TV}} \Bigg| Z_0 = z_0\Bigg] \tag{6.203}$$

$$\leq \mathbb{E}\left[\frac{1}{1-\gamma}\left\|b_{t+1}(\cdot|H_{t+1}) - \hat{b}_{t+1}(\cdot|Z_{t+1})\right\|_{\mathrm{TV}}\middle| Z_0 = z_0\right]. \tag{6.204}$$

Now, by using Lebesgue's dominated theorem in the reverse direction, we have,

$$\mathbb{E}^{\pi_2}\left[\sum_{t=0}^{\infty}\gamma^{t+1}\left(V^{\pi_1}(Z_{t+1}) - \mathcal{V}^{\pi_1}(S_{t+1}, Z_{t+1})\right)\middle| Z_0 = z_0\right]$$

$$\leq \frac{1}{1-\gamma}\mathbb{E}^{\pi_2}\left[\sum_{t=0}^{\infty}\gamma^{t+1}\left\|b_{t+1}(\cdot|H_{t+1}) - \hat{b}_{t+1}(\cdot|Z_{t+1})\right\|_{\mathrm{TV}}\middle| Z_0 = z_0\right] \tag{6.205}$$

$$= \frac{1}{1-\gamma}\mathbb{E}^{\pi_2}\Bigg[\sum_{t=0}^{\infty}\gamma^{t}\left\|b_{t}(\cdot|H_{t}) - \hat{b}_{t}(\cdot|Z_{t})\right\|_{\mathrm{TV}} - \left\|b_0(\cdot|H_0) - \hat{b}_0(\cdot|Z_0)\right\|_{\mathrm{TV}}\Bigg| Z_0 = z_0\Bigg] \tag{6.206}$$

$$= \frac{1}{1-\gamma}\mathbb{E}^{\pi_2}\left[\sum_{t=0}^{\infty}\gamma^{t}\left\|b_{t}(\cdot|H_{t}) - \hat{b}_{t}(\cdot|Z_{t})\right\|_{\mathrm{TV}}\middle| Z_0 = z_0\right] - \mathbb{E}^{\pi_2}\left[\left\|b_0(\cdot|H_0) - \hat{b}_0(\cdot|Z_0)\right\|_{\mathrm{TV}}\middle| Z_0 = z_0\right] \tag{6.207}$$

$$= \frac{1}{1-\gamma}\varepsilon_{\mathrm{inf}}^{\pi_2}(z_0) - \mathbb{E}^{\pi_2}\left[\left\|b_0(\cdot|H_0) - \hat{b}_0(\cdot|Z_0)\right\|_{\mathrm{TV}}\middle| Z_0 = z_0\right] \tag{6.208}$$

$$\leq \frac{1}{1-\gamma}\varepsilon_{\mathrm{inf}}^{\pi_2}(z_0). \tag{6.209}$$

Finally, by substituting the upper bound (6.193) on the first term and the upper bound (6.209) on the second term into equation (6.180), we obtain,

$$V^{\pi_2}(z_0) - V^{\pi_1}(z_0)$$

$$\leq \mathbb{E}^{\pi_2}\left[\sum_{t=0}^{\infty}\gamma^{t}A^{\pi_1}(Z_t, A_t)\middle| Z_0 = z_0\right] + \frac{2}{1-\gamma}\varepsilon_{\mathrm{inf}}^{\pi_2}(z_0) \tag{6.210}$$

$$= \frac{1}{1-\gamma}\mathbb{E}^{d^{\pi_2}}\left[A^{\pi_1}(Z, A)|Z_0 = z_0\right] + \frac{2}{1-\gamma}\varepsilon_{\mathrm{inf}}^{\pi_2}(z_0). \tag{6.211}$$

This concludes the proof. □

Using Lemma 6.E.1, we can prove Theorem 6.5, that is recalled below. The proof from Cayci et al. [2024] is generalized to the asymmetric setting.

Theorem 6.5 (Finite-time bound for asymmetric and symmetric natural actor-critic algorithm). For any agent-state process $\mathcal{M} = (\mathcal{Z}, U)$, we have for Algorithm 6.2 with $\alpha = \frac{1}{\sqrt{K}}$, $\zeta = \frac{B\sqrt{1-\gamma}}{\sqrt{2N}}$, $\eta = \frac{1}{\sqrt{T}}$ and arbitrary $B > 0$,

$$(1-\gamma)\min_{0\leq t<T}\mathbb{E}\left[J(\pi^*) - J(\pi_t)\right] \leq \varepsilon_{\mathrm{nac}} + 2\varepsilon_{\mathrm{inf}}$$

$$+ \overline{C}_\infty \left(\varepsilon_{\text{actor}} + 2\varepsilon_{\text{grad}} + 2\sqrt{6}\frac{1}{T}\sum_{t=0}^{T-1} \varepsilon_{\text{critic}}^{\pi_t} \right), \tag{6.41}$$

where the different terms may differ for asymmetric and symmetric critics,

$$\varepsilon_{\text{nac}} = \frac{B^2 + 2\log|\mathcal{A}|}{2\sqrt{T}} \tag{6.42}$$

$$\varepsilon_{\text{actor}} = \sqrt{\frac{(2-\gamma)B}{(1-\gamma)\sqrt{N}}} \tag{6.43}$$

$$\varepsilon_{\text{inf,asym}} = 0 \tag{6.44}$$

$$\varepsilon_{\text{inf,sym}} = \mathbb{E}^{\pi^*}\left[\sum_{k=0}^{\infty} \gamma^k \left\| \hat{b}_k - b_k \right\|_{\text{TV}} \right] \tag{6.45}$$

$$\varepsilon_{\text{grad,asym}} = \sup_{0\le t<T} \sqrt{\min_w \mathcal{L}_t(w)} \tag{6.46}$$

$$\varepsilon_{\text{grad,sym}} = \sup_{0\le t<T} \sqrt{\min_w L_t(w)}, \tag{6.47}$$

and $\varepsilon_{\text{critic}}^{\pi_t}$ is given in Theorem 6.3 and Theorem 6.4.

Proof. The proof is based on a Lyapounov drift result using the following Lyapounov function,

$$\Lambda(\pi) = \sum_{z\in\mathcal{Z}} d^{\pi^*}(z)\text{KL}(\pi^*(\cdot|z) \parallel \pi(\cdot|z)). \tag{6.212}$$

The Lyapounov drift is given by,

$$\Lambda(\pi_{t+1}) - \Lambda(\pi_t) = \sum_{z\in\mathcal{Z}} d^{\pi^*}(z) \sum_{a\in\mathcal{A}} \pi^*(a|z) \log \frac{\pi_t(a|z)}{\pi_{t+1}(a|z)} \tag{6.213}$$

$$= \sum_{z,a} d^{\pi^*}(z,a) \log \frac{\pi_t(a|z)}{\pi_{t+1}(a|z)}. \tag{6.214}$$

Since $\sup_{z,a} \|\psi(z,a)\|_2 \le 1$, we have that $\log \pi_\theta(a|z)$ is 1-smooth [Agarwal et al., 2021], which implies,

$$\log \pi_{\theta_2}(a|z) \le \log \pi_{\theta_1}(a|z) + \langle \nabla_\theta \log \pi_{\theta_1}(a|z), \theta_2 - \theta_1 \rangle + \frac{1}{2}\|\theta_2 - \theta_1\|_2^2. \tag{6.215}$$

By selecting $\theta_2 = \theta_t$ and $\theta_1 = \theta_{t+1}$ and noting that $\theta_{t+1} - \theta_t = \eta\bar{w}_t = \eta\frac{1}{N}\sum_{n=0}^{N-1} w_{t,n}$ we obtain,

$$\log \frac{\pi_t(a|z)}{\pi_{t+1}(a|z)} \le \frac{\eta^2}{2}\|\bar{w}_t\|_2^2 - \eta\langle \nabla_\theta \log \pi_t(a|z), \bar{w}_t \rangle. \tag{6.216}$$

Now, we separately bound the Lyapounov drift for the asymmetric and symmetric settings. In the following, some notations are overloaded across both setting when their meaning is clear from context. For the asymmetric setting, we have,

$$\Lambda(\pi_{t+1}) - \Lambda(\pi_t)$$

$$= \sum_{z,a} d^{\pi^*}(z,a) \log \frac{\pi_t(a|z)}{\pi_{t+1}(a|z)} \tag{6.217}$$

$$\leq \frac{\eta^2}{2} \|\bar{w}_t\|_2^2 - \eta \sum_{z,a} d^{\pi^*}(z,a) \langle \nabla_\theta \log \pi_t(a|z), \bar{w}_t \rangle \tag{6.218}$$

$$= \frac{\eta^2}{2} B^2 - \eta \sum_{s,z,a} d^{\pi^*}(s,z,a) \mathcal{A}^{\pi_t}(s,z,a) - \eta \sum_{s,z,a} d^{\pi^*}(s,z,a) \left(\langle \nabla_\theta \log \pi_t(a|z), \bar{w}_t \rangle - \mathcal{A}^{\pi_t}(s,z,a) \right) \tag{6.219}$$

$$\leq \frac{\eta^2}{2} B^2 - \eta \sum_{s,z,a} d^{\pi^*}(s,z,a) \mathcal{A}^{\pi_t}(s,z,a) + \eta \sum_{z,a} d^{\pi^*}(s,z,a) \sqrt{\left(\langle \nabla_\theta \log \pi_t(a|z), \bar{w}_t \rangle - \mathcal{A}^{\pi_t}(s,z,a) \right)^2}. \tag{6.220}$$

For the symmetric setting, we observe instead,

$$\Lambda(\pi_{t+1}) - \Lambda(\pi_t)$$

$$= \sum_{z,a} d^{\pi^*}(z,a) \log \frac{\pi_t(a|z)}{\pi_{t+1}(a|z)} \tag{6.221}$$

$$\leq \frac{\eta^2}{2} B^2 - \eta \sum_{z,a} d^{\pi^*}(z,a) A^{\pi_t}(z,a) + \eta \sum_{z,a} d^{\pi^*}(z,a) \sqrt{\left(\langle \nabla_\theta \log \pi_t(a|z), \bar{w}_t \rangle - A^{\pi_t}(z,a) \right)^2}. \tag{6.222}$$

Now, let $\mathfrak{H}_t$ denote the sigma field of all samples used in the computation of π_t (which excludes the samples used for computing $\bar{w}_t$), along with all the samples used in the computation of $\overline{Q}^{\pi_t}$. We define the ideal and approximate loss functions, both in the asymmetric and the symmetric setting,

$$\mathcal{L}_t(w) = \mathbb{E}\left[\left(\langle \nabla_\theta \log \pi_t(A|Z), w \rangle - \mathcal{A}^{\pi_t}(S,Z,A) \right)^2 \middle| \mathfrak{H}_t \right] \tag{6.223}$$

$$\overline{\mathcal{L}}_t(w) = \mathbb{E}\left[\left(\langle \nabla_\theta \log \pi_t(A|Z), w \rangle - \overline{\mathcal{A}}^{\pi_t}(S,Z,A) \right)^2 \middle| \mathfrak{H}_t \right] \tag{6.224}$$

$$L_t(w) = \mathbb{E}\left[\left(\langle \nabla_\theta \log \pi_t(A|Z), w \rangle - A^{\pi_t}(Z,A) \right)^2 \middle| \mathfrak{H}_t \right] \tag{6.225}$$

$$\overline{L}_t(w) = \mathbb{E}\left[\left(\langle \nabla_\theta \log \pi_t(A|Z), w \rangle - \overline{A}^{\pi_t}(Z,A) \right)^2 \middle| \mathfrak{H}_t \right]. \tag{6.226}$$

Because $\mathbb{E}\left[\left\| \mathcal{V}^{\pi_t} - \overline{\mathcal{V}}^{\pi_t} \right\|_{d^{\pi_t}}^2 \middle| \mathfrak{H}_t \right] \leq \mathbb{E}\left[\left\| \overline{\mathcal{Q}}^{\pi_t} - \mathcal{Q}^{\pi_t} \right\|_{d^{\pi_t}}^2 \middle| \mathfrak{H}_t \right]$, the error between the asymmetric advantage $\mathcal{A}$ and its approximation $\bar{\mathcal{A}}$ is upper bounded by,

$$\sqrt{\mathbb{E}\left[\left(\overline{\mathcal{A}}^{\pi_t}(S,Z,A) - \mathcal{A}^{\pi_t}(S,Z,A) \right)^2 \middle| \mathfrak{H}_t \right]}$$

$$= \sqrt{\mathbb{E}\left[\left\| \overline{\mathcal{A}}^{\pi_t} - \mathcal{A}^{\pi_t} \right\|_{d^{\pi_t}}^2 \middle| \mathfrak{H}_t \right]} \tag{6.227}$$

$$= \sqrt{\mathbb{E}\left[\left\| \overline{\mathcal{Q}}^{\pi_t} - \overline{\mathcal{V}}^{\pi_t} - \mathcal{Q}^{\pi_t} + \mathcal{V}^{\pi_t} \right\|_{d^{\pi_t}}^2 \middle| \mathfrak{H}_t \right]} \tag{6.228}$$

$$= \sqrt{\mathbb{E}\left[\left\|\overline{\mathcal{Q}}^{\pi_t} - \mathcal{Q}^{\pi_t} + \mathcal{V}^{\pi_t} - \overline{\mathcal{V}}^{\pi_t}\right\|_{d^{\pi_t}}^2 \middle| \mathfrak{H}_t\right]} \tag{6.229}$$

$$\leq \sqrt{\mathbb{E}\left[\left\|\overline{\mathcal{Q}}^{\pi_t} - \mathcal{Q}^{\pi_t}\right\|_{d^{\pi_t}}^2 + \left\|\mathcal{V}^{\pi_t} - \overline{\mathcal{V}}^{\pi_t}\right\|_{d^{\pi_t}}^2 \middle| \mathfrak{H}_t\right]} \tag{6.230}$$

$$\leq \sqrt{\mathbb{E}\left[\left\|\overline{\mathcal{Q}}^{\pi_t} - \mathcal{Q}^{\pi_t}\right\|_{d^{\pi_t}}^2 \middle| \mathfrak{H}_t\right]} + \sqrt{\mathbb{E}\left[\left\|\mathcal{V}^{\pi_t} - \overline{\mathcal{V}}^{\pi_t}\right\|_{d^{\pi_t}}^2 \middle| \mathfrak{H}_t\right]} \tag{6.231}$$

$$\leq 2\varepsilon_{\text{critic,asym}}^{\pi_t}, \tag{6.232}$$

where $\varepsilon_{\text{critic,asym}}^{\pi_t} = \varepsilon_{\text{td,asym}}^{\pi_t} + \varepsilon_{\text{app,asym}}^{\pi_t} + \varepsilon_{\text{shift,asym}}^{\pi_t}$ is given by the upper bound (6.29) in Theorem 6.3. Similarly, the error between the symmetric advantage A and its approximation $\overline{A}$ is upper bounded by,

$$\sqrt{\mathbb{E}\left[\left(\overline{A}^{\pi_t}(Z,A) - A^{\pi_t}(Z,A)\right)^2 \middle| \mathfrak{H}_t\right]} \leq 2\varepsilon_{\text{critic,sym}}^{\pi_t}, \tag{6.233}$$

where $\varepsilon_{\text{critic,sym}}^{\pi_t} = \varepsilon_{\text{td,sym}}^{\pi_t} + \varepsilon_{\text{app,sym}}^{\pi_t} + \varepsilon_{\text{shift,sym}}^{\pi_t} + \varepsilon_{\text{alias,sym}}^{\pi_t}$ is given by the upper bound (6.35) in Theorem 6.4. By using the inequality $(x+y)^2 \leq 2x^2 + 2y^2$,

$$\overline{\mathcal{L}}_t(w) = \mathbb{E}\left[\left(\langle \nabla_\theta \log \pi_t(A|Z), w\rangle - \overline{\mathcal{A}}^{\pi_t}(S,Z,A)\right)^2 \middle| \mathfrak{H}_t\right] \tag{6.234}$$

$$= \mathbb{E}\Bigg[\Big(\langle \nabla_\theta \log \pi_t(A|Z), w\rangle - \mathcal{A}^{\pi_t}(S,Z,A) + \mathcal{A}^{\pi_t}(S,Z,A) - \overline{\mathcal{A}}^{\pi_t}(S,Z,A)\Big)^2 \Bigg| \mathfrak{H}_t\Bigg] \tag{6.235}$$

$$\leq 2\mathbb{E}\left[\left(\langle \nabla_\theta \log \pi_t(A|Z), w\rangle - \mathcal{A}^{\pi_t}(S,Z,A)\right)^2 \middle| \mathfrak{H}_t\right] + 2\mathbb{E}\left[\left(\mathcal{A}^{\pi_t}(S,Z,A) - \overline{\mathcal{A}}^{\pi_t}(S,Z,A)\right)^2 \middle| \mathfrak{H}_t\right] \tag{6.236}$$

$$\leq 2\mathcal{L}_t(w) + 2(2\varepsilon_{\text{critic,asym}}^{\pi_t})^2. \tag{6.237}$$

Similarly, we obtain in the symmetric case,

$$\overline{L}_t(w) \leq 2L_t(w) + 2(2\varepsilon_{\text{critic,sym}}^{\pi_t})^2. \tag{6.238}$$

Starting from the ideal objective and following a similar technique, we also obtain,

$$\mathcal{L}_t(w) \leq 2\overline{\mathcal{L}}_t(w) + 2(2\varepsilon_{\text{critic,asym}}^{\pi_t})^2 \tag{6.239}$$

$$L_t(w) \leq 2\overline{L}_t(w) + 2(2\varepsilon_{\text{critic,sym}}^{\pi_t})^2. \tag{6.240}$$

By using Theorem 14.8 in [Shalev-Shwartz and Ben-David, 2014] with step size $\zeta = \frac{B\sqrt{1-\gamma}}{\sqrt{2N}}$, we obtain for the average iterate $\bar{w}_t$ under the asymmetric loss and symmetric loss, respectively,

$$\overline{\mathcal{L}}_t(\bar{w}_t) \leq \varepsilon_{\text{actor}}^2 + \min_{\|w\|_2 \leq B} \overline{\mathcal{L}}_t(w) \tag{6.241}$$

$$\overline{L}_t(\bar{w}_t) \leq \varepsilon_{\text{actor}}^2 + \min_{\|w\|_2 \leq B} \overline{L}_t(w), \tag{6.242}$$

where $\varepsilon^2_{\text{actor}} = \frac{(2-\gamma)B}{2(1-\gamma)\sqrt{N}}$. On expectation, for the ideal asymmetric objective $\mathcal{L}_t$, we obtain,

$$\mathbb{E}\left[\mathcal{L}_t(\bar{w}_t)\right] \leq 2\mathbb{E}\left[\bar{\mathcal{L}}_t(\bar{w}_t)\right] + 2(2\varepsilon^{\pi_t}_{\text{critic,asym}})^2 \tag{6.243}$$

$$\leq 2\varepsilon^2_{\text{actor}} + 2\min_{\|w\|_2 \leq B} \bar{\mathcal{L}}_t(w) + 2(2\varepsilon^{\pi_t}_{\text{critic,asym}})^2 \tag{6.244}$$

$$\leq 2\varepsilon^2_{\text{actor}} + 2\left(2\min_{\|w\|_2 \leq B} \mathcal{L}_t(w) + 2(2\varepsilon^{\pi_t}_{\text{critic,asym}})^2\right) + 2(2\varepsilon^{\pi_t}_{\text{critic,asym}})^2 \tag{6.245}$$

$$= 2\varepsilon^2_{\text{actor}} + 4\min_{\|w\|_2 \leq B} \mathcal{L}_t(w) + 6(2\varepsilon^{\pi_t}_{\text{critic,asym}})^2 \tag{6.246}$$

$$= 2\varepsilon^2_{\text{actor}} + 4\left(\varepsilon^{\pi_t}_{\text{grad,asym}}\right)^2 + 6(2\varepsilon^{\pi_t}_{\text{critic,asym}})^2, \tag{6.247}$$

where we define the actor gradient function approximation error as,

$$\left(\varepsilon^{\pi_t}_{\text{grad,asym}}\right)^2 = \min_{\|w\|_2 \leq B} \mathcal{L}_t(w). \tag{6.248}$$

Similarly, we obtain on expectation for the ideal symmetric objective L_t,

$$\mathbb{E}\left[L_t(\bar{w}_t)\right] \leq 2\varepsilon^2_{\text{actor}} + 4\left(\varepsilon^{\pi_t}_{\text{grad,sym}}\right)^2 + 6(2\varepsilon^{\pi_t}_{\text{critic,sym}})^2, \tag{6.249}$$

where we define the actor gradient function approximation error as,

$$\left(\varepsilon^{\pi_t}_{\text{grad,sym}}\right)^2 = \min_{\|w\|_2 \leq B} L_t(w). \tag{6.250}$$

Now, let us go back to the asymmetric and symmetric Lyapounov drift functions of equation (6.220) and (6.222). First, we assume that there exists $\overline{C}_\infty < \infty$ such that $\sup_{t \geq 0} \mathbb{E}[C_t] \leq \overline{C}_\infty$ with,

$$C_t = \sup_{s,z,a} \left| \frac{d^{\pi^*}(s,z)\pi^*(a|z)}{d^{\pi_{\theta_t}}(s,z)\pi_{\theta_t}(a|z)} \right|. \tag{6.251}$$

Second, we leverage the performance difference lemma to bound the advantage. For the asymmetric setting, the performance difference lemma for MDP [Kakade and Langford, 2002] holds because of the Markovianity of (S_t, Z_t),

$$(1-\gamma)\left(V^{\pi^*}(s_0,z_0) - V^{\pi_t}(s_0,z_0)\right) = \mathbb{E}^{d^{\pi^*}}\left[\mathcal{A}^{\pi_t}(S,Z,A)|S_0 = s_0, Z_0 = z_0\right]. \tag{6.252}$$

We note that $\mathbb{E}\left[V^{\pi^*}(S_0,Z_0) - V^{\pi_t}(S_0,Z_0)\right] = \mathbb{E}\left[J(\pi^*) - J(\pi_t)\right]$, such that,

$$-\mathbb{E}^{d^{\pi^*}}\left[\mathcal{A}^{\pi_t}(S,Z,A)\right] = -(1-\gamma)\left(J(\pi^*) - J(\pi_t)\right). \tag{6.253}$$

$$= -(1-\gamma)\left(J(\pi^*) - J(\pi_t)\right) + \varepsilon_{\text{inf,asym}}, \tag{6.254}$$

where $\varepsilon_{\text{inf,asym}} = 0$. For the symmetric setting, using Lemma 6.E.1 with $\pi_2 = \pi^*$ and $\pi_1 = \pi_t$, we note that,

$$(1-\gamma)\left(V^{\pi^*}(z_0) - V^{\pi_t}(z_0)\right) \leq \mathbb{E}^{d^{\pi^*}}\left[A^{\pi_t}(Z,A)|Z_0 = z_0\right] + 2\varepsilon^{\pi^*}_{\text{inf}}(z_0), \tag{6.255}$$

which implies,

$$-\mathbb{E}^{d^{\pi^*}}\left[A^{\pi_t}(Z,A)|Z_0=z_0\right] \leq -(1-\gamma)\left(V^{\pi^*}(z_0)-V^{\pi_t}(z_0)\right)+2\varepsilon_{\text{inf}}^{\pi^*}(z_0). \tag{6.256}$$

We note that $\mathbb{E}\left[V^{\pi^*}(Z_0)-V^{\pi_t}(Z_0)\right]=\mathbb{E}\left[J(\pi^*)-J(\pi_t)\right]$ and we denote $\mathbb{E}\left[\varepsilon_{\text{inf}}^{\pi^*}(Z_0)\right]$ with $\varepsilon_{\text{inf,sym}}$, so that,

$$\varepsilon_{\text{inf,sym}} = \mathbb{E}\left[\mathbb{E}^{\pi^*}\left[\sum_{k=0}^{\infty}\gamma^k\left\|\hat{b}_k-b_k\right\|_{\text{TV}}\middle| Z_0=Z_0\right]\right] \tag{6.257}$$

$$= \mathbb{E}^{\pi^*}\left[\sum_{k=0}^{\infty}\gamma^k\left\|\hat{b}_k-b_k\right\|_{\text{TV}}\right]. \tag{6.258}$$

By rearranging, we have,

$$-\mathbb{E}^{d^{\pi^*}}\left[A^{\pi_t}(Z,A)\right] \leq -(1-\gamma)\mathbb{E}\left[J(\pi^*)-J(\pi_t)\right]+2\varepsilon_{\text{inf,sym}}. \tag{6.259}$$

Note that $\sum_{s,z,a} d^{\pi^*}(s,z,a)f(s,z,a) = \sum_{s,z,a}\frac{d^{\pi^*}(s,z,a)}{d^{\pi_t}(s,z,a)}d^{\pi_t}(s,z,a)f(s,z,a) \leq C_t\sum_{s,z,a} d^{\pi_t}(s,z,a)f(s,z,a)$ for positive f. Taking expectation over the asymmetric Lyapounov drift of equation (6.220), we obtain using equation (6.251),

$$\begin{aligned}
&\mathbb{E}\left[\Lambda(\pi_{t+1})-\Lambda(\pi_t)\right] \\
&\quad\leq \frac{\eta^2}{2}B^2-\eta\sum_{z,a}d^{\pi^*}(z,a)A^{\pi_t}(z,a) \\
&\qquad+\eta\sum_{s,z,a}d^{\pi^*}(s,z,a)\sqrt{\left(\langle\nabla_\theta\log\pi_t(a|z),\bar{w}_t\rangle-\mathcal{A}^{\pi_t}(s,z,a)\right)^2} &(6.260)\\
&\quad\leq \frac{\eta^2}{2}B^2-\eta(1-\gamma)\mathbb{E}\left[J(\pi^*)-J(\pi_t)\right]+2\eta\varepsilon_{\text{inf,asym}} \\
&\qquad+\eta\overline{C}_\infty\sqrt{2\varepsilon_{\text{actor}}^2+4\left(\varepsilon_{\text{grad,asym}}^{\pi_t}\right)^2+6(2\varepsilon_{\text{critic,asym}}^{\pi_t})^2} &(6.261)\\
&\quad\leq \frac{\eta^2}{2}B^2-\eta(1-\gamma)\mathbb{E}\left[J(\pi^*)-J(\pi_t)\right]+2\eta\varepsilon_{\text{inf,asym}} \\
&\quad+\eta\overline{C}_\infty\left(\sqrt{2}\varepsilon_{\text{actor}}+2\varepsilon_{\text{grad,asym}}^{\pi_t}+2\sqrt{6}\varepsilon_{\text{critic,asym}}^{\pi_t}\right). &(6.262)
\end{aligned}$$

Similarly, taking expectation over the symmetric drift of equation (6.222), we obtain a similar expression,

$$\begin{aligned}
&\mathbb{E}\left[\Lambda(\pi_{t+1})-\Lambda(\pi_t)\right] \\
&\quad\leq \frac{\eta^2}{2}B^2-\eta\sum_{z,a}d^{\pi^*}(z,a)A^{\pi_t}(z,a) \\
&\qquad+\eta\sum_{z,a}d^{\pi^*}(z,a)\sqrt{\left(\langle\nabla_\theta\log\pi_t(a|z),\bar{w}_t\rangle-A^{\pi_t}(z,a)\right)^2} &(6.263)\\
&\quad\leq \frac{\eta^2}{2}B^2-\eta(1-\gamma)\mathbb{E}\left[J(\pi^*)-J(\pi_t)\right]+2\eta\varepsilon_{\text{inf,sym}}
\end{aligned}$$

$$+ \eta \overline{C}_\infty \left(\sqrt{2}\varepsilon_{\text{actor}} + 2\varepsilon^{\pi_t}_{\text{grad,sym}} + 2\sqrt{6}\varepsilon^{\pi_t}_{\text{critic,sym}} \right). \tag{6.264}$$

Given the similarity of equation (6.262) and equation (6.264), in the following we denote the denote the upper bounds using ε_{inf}, $\varepsilon^{\pi_t}_{\text{grad}}$ and $\varepsilon^{\pi_t}_{\text{critic}}$, irrespectively of the setting (i.e., asymmetric or symmetric).

By summing all Laypounov drifts, we obtain,

$$\begin{aligned}
&\mathbb{E}\left[\Lambda(\pi_T) - \Lambda(\pi_0)\right] \\
&\quad \leq T\frac{\eta^2}{2}B^2 - \eta(1-\gamma)\sum_{t=0}^{T-1}\mathbb{E}\left[J(\pi^*) - J(\pi_t)\right] + 2\eta T\varepsilon_{\text{inf}} \\
&\qquad + \eta\sum_{t=0}^{T-1}\overline{C}_\infty\left(\sqrt{2}\varepsilon_{\text{actor}} + 2\varepsilon^{\pi_t}_{\text{grad}} + 2\sqrt{6}\varepsilon^{\pi_t}_{\text{critic}}\right) && (6.265) \\
&\quad \leq T\frac{\eta^2}{2}B^2 - \eta(1-\gamma)\sum_{t=0}^{T-1}\mathbb{E}\left[J(\pi^*) - J(\pi_t)\right] + 2\eta T\varepsilon_{\text{inf}} \\
&\qquad + \eta\overline{C}_\infty\left(\sqrt{2}T\varepsilon_{\text{actor}} + 2\sum_{t=0}^{T-1}\varepsilon^{\pi_t}_{\text{grad}} + 2\sqrt{6}\sum_{t=0}^{T-1}\varepsilon^{\pi_t}_{\text{critic}}\right). && (6.266)
\end{aligned}$$

Since π_0 is initialized at the uniform policy with $\theta_0 := 0$, we have,

$$\begin{aligned}
\Lambda(\pi_0) &= \sum_{z\in\mathcal{Z}} d^{\pi^*}(z)\text{KL}(\pi^*(\cdot|z) \parallel \pi_0(\cdot|z)) && (6.267) \\
&= \sum_{z\in\mathcal{Z}} d^{\pi^*}(z)\left(\sum_{a\in\mathcal{A}}\pi^*(a|z)\log\pi^*(a|z) - \sum_{a\in\mathcal{A}}\pi^*(a|z)\log\pi_0(a|z)\right) && (6.268) \\
&= \sum_{z\in\mathcal{Z}} d^{\pi^*}(z)\left(\sum_{a\in\mathcal{A}}\pi^*(a|z)\log\pi^*(a|z) - \sum_{a\in\mathcal{A}}\pi^*(a|z)\log\frac{1}{|\mathcal{A}|}\right) && (6.269) \\
&= \sum_{z\in\mathcal{Z}} d^{\pi^*}(z)\left(\sum_{a\in\mathcal{A}}\pi^*(a|z)\log\pi^*(a|z) + \log|\mathcal{A}|\right) && (6.270) \\
&= \sum_{z\in\mathcal{Z}} d^{\pi^*}(z)\left(\log|\mathcal{A}| - H(\pi^*(\cdot|z))\right) && (6.271) \\
&\leq \sum_{z\in\mathcal{Z}} d^{\pi^*}(z)\log|\mathcal{A}| && (6.272) \\
&\leq \log|\mathcal{A}|, && (6.273)
\end{aligned}$$

where H denotes the Shannon entropy. Rearranging and dividing by ηT, we obtain after neglecting $\mathcal{L}(\pi_T) > 0$,

$$\begin{aligned}
(1-\gamma)\frac{1}{T}\sum_{t=0}^{T-1}\mathbb{E}\left[J(\pi^*) - J(\pi_t)\right] &\leq \frac{\log|\mathcal{A}|}{\eta T} + \frac{\eta}{2}B^2 + 2\varepsilon_{\text{inf}} \\
&\quad + \overline{C}_\infty\left(\sqrt{2}\varepsilon_{\text{actor}} + 2\frac{1}{T}\sum_{t=0}^{T-1}\varepsilon^{\pi_t}_{\text{grad}} + 2\sqrt{6}\frac{1}{T}\sum_{t=0}^{T-1}\varepsilon^{\pi_t}_{\text{critic}}\right). && (6.274)
\end{aligned}$$

It can also be noted that $\min_{0\leq t<T}[x_t] \leq \frac{1}{T}\sum_{t=0}^{T} x_t$, which implies that,

$$(1-\gamma)\min_{0\leq t<T}\mathbb{E}\left[J(\pi^*)-J(\pi_t)\right] \leq \frac{\log|\mathcal{A}|}{\eta T}+\frac{\eta}{2}B^2+2\varepsilon_{\text{inf}} + \overline{C}_{\infty}\left(\sqrt{2}\varepsilon_{\text{actor}}+2\frac{1}{T}\sum_{t=0}^{T-1}\varepsilon_{\text{grad}}^{\pi_t}+2\sqrt{6}\frac{1}{T}\sum_{t=0}^{T-1}\varepsilon_{\text{critic}}^{\pi_t}\right). \tag{6.275}$$

Let us define the worse actor gradient function approximation error,

$$\varepsilon_{\text{grad}} = \sup_{0\leq t<T}\varepsilon_{\text{grad}}^{\pi_t} \tag{6.276}$$

$$= \sup_{0\leq t<T}\sqrt{\min_{\|w\|_2\leq B} L_t(w)}, \tag{6.277}$$

and let us note that,

$$\frac{1}{T}\sum_{t=0}^{T-1}\varepsilon_{\text{grad}}^{\pi_t} \leq \varepsilon_{\text{grad}}. \tag{6.278}$$

By setting $\eta=\frac{1}{\sqrt{T}}$, we obtain,

$$(1-\gamma)\min_{0\leq t<T}\mathbb{E}\left[J(\pi^*)-J(\pi_t)\right] \leq \frac{\log|\mathcal{A}|}{\sqrt{T}}+\frac{B^2}{2\sqrt{T}}+2\varepsilon_{\text{inf}} + \overline{C}_{\infty}\left(\sqrt{2}\varepsilon_{\text{actor}}+2\frac{1}{T}\sum_{t=0}^{T-1}\varepsilon_{\text{grad}}^{\pi_t}+2\sqrt{6}\frac{1}{T}\sum_{t=0}^{T-1}\varepsilon_{\text{critic}}^{\pi_t}\right) \tag{6.279}$$

$$= \frac{B^2+2\log|\mathcal{A}|}{2\sqrt{T}}+2\mathbb{E}^{\pi^*}\left[\sum_{k=0}^{\infty}\gamma^k\left\|\hat{b}_k-b_k\right\|_{\text{TV}}\right] + \overline{C}_{\infty}\left(\sqrt{\frac{(2-\gamma)B}{(1-\gamma)\sqrt{N}}}+2\varepsilon_{\text{grad}}+2\sqrt{6}\frac{1}{T}\sum_{t=0}^{T-1}\varepsilon_{\text{critic}}^{\pi_t}\right). \tag{6.280}$$

This concludes the proof. □

Part III

Entangling Predictions and Decisions

Entangling Predictions and Decisions

In this part, we introduce several architectural and algorithmic contributions that result in an efficient generative sequence model. First, we introduce a latent generative model based on parallelizable linear recurrent neural networks, allowing parallel generation. We also show that this model presents several other advantages such as its implicit recurrence, which allows resuming generation without reprocessing the past. Second, we discuss the application of such world models in combination with latent policies, for anticipating and learning from the future without explicitly modeling it. More precisely, we introduce a world model that allows parallel imagination using such latent policies.

Chapter 7

Rolling the Dice First

Parallelizing Autoregressive Generation with Variational State Space Models. Gaspard Lambrechts, Yann Claes, Pierre Geurts and Damien Ernst.

From the paper presented at the *ICML Workshop on the Next Generation of Sequence Modeling Architectures.*

Abstract

Attention-based models such as Transformers and recurrent models like state space models (SSM) have emerged as successful methods for autoregressive sequence modeling. Although both enable parallel training, none enable parallel generation due to their autoregressiveness. We propose the variational SSM (VSSM), a variational autoencoder (VAE) where both the encoder and decoder are SSMs. Since sampling the latent variables and decoding them with the SSM can be parallelized, both training and generation can be conducted in parallel. Moreover, the decoder recurrence allows generation to be resumed without reprocessing the whole sequence. Finally, we propose the autoregressive VSSM that can be conditioned on a partial realization of the sequence, as is common in language generation tasks. Interestingly, the autoregressive VSSM still enables parallel generation. We highlight on toy problems (MNIST, CIFAR) the empirical gains in speed-up and show that it competes with traditional models in terms of generation quality (Transformer, Mamba SSM).

7.1 Introduction

Sequence modeling tasks, namely time-series forecasting and text generation, have gained in popularity and various types of architectures were designed to tackle such problems. Transformers were proven effective [Vaswani et al., 2017, Radford et al., 2019], yet they nonetheless reprocess the complete sequence at each time step, making generation less efficient. Recurrent neural networks (RNN) [Graves, 2013, Cho et al., 2014b] update a hidden state based on new inputs at each time step, enabling efficient generation. SSMs [Gupta et al., 2022, Gu et al., 2022a, Smith et al., 2023, Gu and Dao, 2023], a recently introduced class of RNNs, enable parallel training thanks to their linear recurrence. Alternatively, several works adapt VAEs for sequential modeling. Some architectures integrate Transformers [Liu and Liu, 2019, Jiang et al., 2020] and enable parallel training, although little work [Fang et al., 2021] proposes models that can be conditioned on partial realizations (e.g., prompts). Conversely, variational RNNs (VRNN) [Chung et al., 2015] loose parallelizability by making the model both autoregressive and recurrent, allowing it to be conditioned on partial realizations and to resume generation. However, all introduced autoregressive models perform generation sequentially, as they are explicitly conditioned on previously generated data.

Therefore, we propose the VSSM, a VAE whose encoder and decoder are SSMs. Thanks to key architectural choices, both training and inference can be performed in parallel and linear time with respect to the sequence length, while still allowing generation to be to resumed without reprocessing the entire sequence. In contrast, a VAE with Transformer encoder and decoder, which we call Transformer VAE (TVAE), would preserve parallel training and generation, but would not be resumable. We then propose the autoregressive VSSM, that can be conditioned on partial realizations of the sequence and still generates in parallel. The VSSM combines all advantages of previous models, as observed in Figure 7.1a, while producing results comparable to Transformers and SSMs on simple tasks (MNIST, CIFAR). We highlight a recent work [Zhou et al., 2023] that proposes a similar architecture, yet their prior and generative models are explicitly autoregressive and do not exploit the parallelizability of SSMs. Moreover, they only consider generation from sampled latents, while we also propose an approach to condition the model on partial realizations. We do not consider diffusion models for sequences (e.g., [Gong et al., 2023]), but note that they would not allow recurrent (i.e., resuming) generation.

7.2 Background

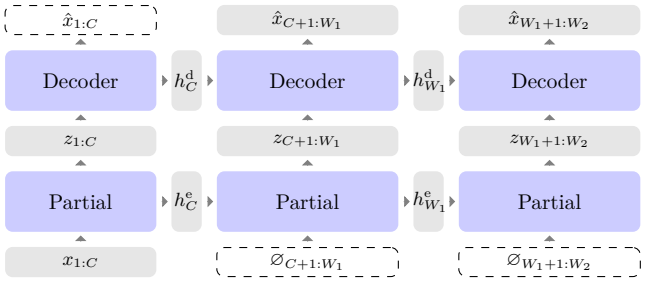
7.2.1 Variational Autoencoders for Time Series

We consider dynamical VAEs [Girin et al., 2021], that model sequential data $x_{1:T}$ of length T through T latent variables $z_{1:T}$. Given a target space $\mathcal{X}$, they define the joint distribution $p_\phi(x_{1:T}, z_{1:T})$ with,

- A latent space $\mathcal{Z}$,
- A prior distribution $p_\phi(z_{1:T}) = \prod_{t=1}^{T} p_\phi(z_t|z_{1:t-1})$,

Model	Training	//	Sampling	//	Prompt	Resume
Transformer	$O(T^2)$	✓	$O(T^2)$	✗	✓	✗
RNN	$O(T)$	✗	$O(T)$	✗	✓	✓
SSM	$O(T)$	✓	$O(T)$	✗	✓	✓
TVAE	$O(T^2)$	✓	$O(T^2)$	✓	✗/✓	✗
VRNN	$O(T)$	✗	$O(T)$	✗	✓	✓
VSSM	$O(T)$	✓	$O(T)$	✓	✓	✓

(a) Time complexities and parallelizability at training time and sampling time, and generation properties.



(b) Parallel and recurrent sampling algorithm, given a contextual prompt $x_{1:C}$.

Figure 7.1: Sequence models properties and variational state space model sampling algorithm.

- A generative distribution $p_\phi(x_{1:T}|z_{1:T}) = \prod_{t=1}^T p_\phi(x_t|x_{1:t-1}, z_{1:T})$,

where ϕ denotes the parameters of these probability distributions. Unfortunately, the likelihood of the data $p_\phi(x_{1:T}) = \mathbb{E}_{p_\phi(z_{1:T})} p_\phi(x_{1:T}|z_{1:T})$ under this model cannot be evaluated in practice. Nevertheless, we can show that the log-likelihood is lower bounded by the evidence lower bound (ELBO), for any conditional probability distribution $q(z_{1:T}|x_{1:T})$,

$$\log p_\phi(x_{1:T}) \geq \mathop{\mathbb{E}}_{q(z_{1:T}|x_{1:T})} \log p_\phi(x_{1:T}|z_{1:T}) - \mathrm{KL}(q(z_{1:T}|x_{1:T}) \parallel p_\phi(z_{1:T})) \quad (7.1)$$

$$= \mathrm{ELBO}_\phi(x_{1:T}). \quad (7.2)$$

Moreover, the ELBO becomes tight when $q(z_{1:T}|x_{1:T})$ corresponds to the true posterior distribution $p_\phi(z_{1:T}|x_{1:T})$. Thus, the generative model p_ϕ is usually jointly optimized with,

- A posterior distribution $q_\psi(z_{1:T}|x_{1:T}) = \prod_{t=1}^T q_\psi(z_t|z_{1:t-1}, x_{1:T})$,

where ψ denotes the parameters of this distribution. These four components compose the dynamical VAE. More details are provided in Appendix 7.A.

7.2.2 State Space Models

SSMs are linear and time-invariant dynamical systems that can be discretized into $h_t = Ah_{t-1} + Bu_t$, where $\zeta = (A, B)$ are learnable parameters. Using the prefix-sum algorithm [Blelloch, 1990], we can parallelize the computation of the

state sequence $h_t = \text{SSM}_\zeta(u_{1:t})$ along all time steps $t \in [1, T]$. Furthermore, we can obtain effective sequence models of the form $y_t = f_\theta(u_{1:t})$ by stacking L layers $i = \{1, \dots, L\}$ of interleaved SSMs and time-step-wise feedforward neural networks (FNN),

$$h_t^i = \text{SSM}_{\zeta_i}(u_{1:t}^{i-1}), \tag{7.3}$$

$$y_t^i = \text{FNN}_{\xi_i}(h_t^i), \tag{7.4}$$

where $u_t^i = y_t^{i-1}$, $u_t^0 = u_t$, $y_t = y_t^L$, and $\theta = \cup_{i=1}^L(\zeta_i, \xi_i)$ includes all SSMs and FNNs parameters. Indeed, it is believed that such stacking of SSMs and time-step-wise FNNs is a universal approximator of sufficiently regular non-linear sequence-to-sequence maps [Orvieto et al., 2023].

7.3 Method

7.3.1 Variational State Space Model

We introduce the VSSM as an instance of dynamical VAE, where we select, given a target space $\mathcal{X}$,

- A discrete latent space of Z components of cardinality N each,

$$\mathcal{Z} = \{1, \dots, N\}^Z, \tag{7.5}$$

- A uniform prior distribution

$$p_\phi(z_{1:T}) = \prod_{t=1}^T p_\phi(z_t|z_{1:t-1}) = \prod_{t=1}^T p_\phi(z_t) = \prod_{t=1}^T \frac{1}{N^Z}, \tag{7.6}$$

- A generative distribution,

$$p_\phi(x_{1:T}|z_{1:T}) = \prod_{t=1}^T p_\phi(x_t|z_{1:t}) = \prod_{t=1}^T \mathcal{P}(x_t|f_\phi^{\text{dec}}(z_{1:t})), \tag{7.7}$$

  where $\mathcal{P}(x_t|w_t)$[1] is a distribution of parameters $w_t = f_\phi^{\text{dec}}(z_{1:t})$ outputted by a stacked SSM,

- A posterior distribution,

$$q_\psi(z_{1:T}|x_{1:T}) = \prod_{t=1}^T q_\psi(z_t|x_{1:t}) = \prod_{t=1}^T \mathcal{D}(z_t|f_\psi^{\text{enc}}(x_{1:t})), \tag{7.8}$$

  where $\mathcal{D}(z_t|v_t)$ is a discrete distribution of probabilities $v_t = f_\psi^{\text{enc}}(x_{1:t})$ outputted by a stacked SSM.

The independence of the prior over all time steps z_t, along with the conditional independence between $z_{\neq t}$ and z_t given $x_{1:t}$ in q_ψ, and between $x_{\neq t}$ and x_t given $z_{1:T}$ in p_ϕ enables the prior, posterior and generative models to be sampled in parallel. Note that the discrete latent space requires the Gumbel reparametrization trick for computing $\nabla_\psi z_{1:T}$ when maximizing the ELBO [Jang et al., 2017, Maddison et al., 2016].

[1] Gaussian of mean w_t and fixed variance for continuous $\mathcal{X}$ or discrete distribution of probabilities w_t for discrete $\mathcal{X}$.

7.3.2 Autoregressive Variational State Space Model

In some applications (e.g., language modeling), it is useful to learn a generative model of the distribution $p(x_{1:T}|x_{1:C})$ conditioned on a partial realization $x_{1:C}$. Under the modeling assumptions of a trained dynamical VAE like the VSSM prior and generative models of Subsection 7.3.1, we have,

$$p_\phi(x_{1:T}|x_{1:C}) = \int_{\mathcal{Z}^T} p_\phi(x_{1:T}|z_{1:T}) p_\phi(z_{1:T}|x_{1:C}) \, \mathrm{d}z_{1:T}, \tag{7.9}$$

where $p_\phi(x_{1:T}|z_{1:T})$ is our generative model, while $p_\phi(z_{1:T}|x_{1:C})$ is the true partial posterior, from which we cannot sample for a given $x_{1:C}$. We thus propose to learn an approximate partial posterior $q_\omega(z_{1:T}|x_{1:C})$ of the true partial posterior $p_\phi(z_{1:T}|x_{1:C})$, by exploiting samples $p(x_{1:T}|x_{1:C})$ from the dataset to construct samples of $p_\phi(z_{1:T}|x_{1:C})$ (see details in Subappendix 7.A.2).

The partial posterior $q_\omega(z_{1:T}|x_{1:C})$ is implemented with a stacked SSM, where the input $x_{1:C}$ is padded with empty tokens: $\bar{x}_{1:T} = (x_{1:C}, \varnothing, \dots, \varnothing)$. The autoregressive VSSM is a VSSM with,

- A partial posterior distribution,

$$q_\omega(z_{1:T}|x_{1:C}) = \prod_{t=1}^{T} q_\omega(z_t|x_{1:\min(C,t)}) = \prod_{t=1}^{T} \mathcal{D}(z_t|f_\omega^{\mathrm{par}}(\bar{x}_{1:t})). \tag{7.10}$$

  where $\mathcal{D}(z_t|\bar{v}_t)$ is a discrete distribution of probabilities $\bar{v}_t = f_\omega^{\mathrm{par}}(\bar{x}_{1:t})$ outputted by a stacked SSM.

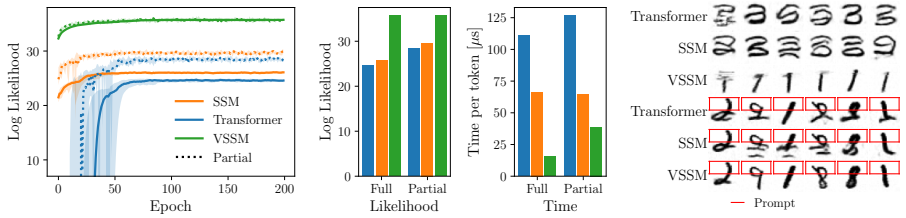
Note that the partial posterior distribution q_ω should ideally correspond to the prior when $\bar{x}_{1:T} = (\varnothing, \dots, \varnothing)$, and it will be used in practice for unconditional generation.

The autoregressive VSSM generates in parallel, possibly conditioned on a partial realization, and can resume generation, as illustrated on Figure 7.1 (see detailed algorithms comparison in Appendix 7.B).

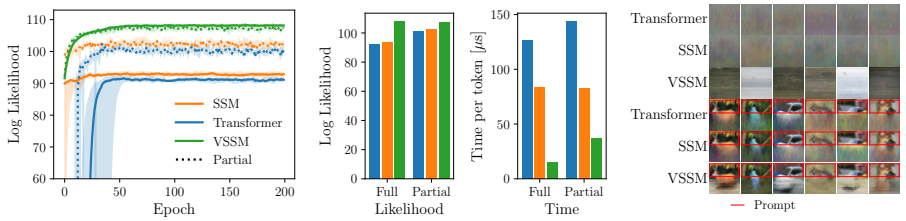
7.4 Experiments

In the following, we compare Transformer, SSM and VSSM on two toy sequence modeling tasks: MNIST, for which we consider 28-dimensional sequences of length 28, and CIFAR, for which we consider (32×3)-dimensional sequences of length 32. Transformer and SSM both output the mean of a Gaussian distribution of fixed variance. For more details about model architectures, see Subappendix 7.C.1. We report samples, generation times and likelihoods in Figure 7.2, estimated by importance-sampling for the VSSM, see Subappendix 7.C.2. We also report additional results in Appendix 7.D. It can be observed that the VSSM arguably outperforms the SSM and Transformer in terms of generation quality. The VSSM also outperforms the other methods in terms of estimated log likelihood on the test set. These results should however be treated with caution, as the log likelihood is a rather crude estimation of the quality of a generative model. Moreover, we also observed in practice that it was very sensitive to the standard deviation parameters are selected for the generative

distribution $p_\phi(x_{1:T}|z_{1:T})$, for all three models. Finally, as expected by the parallelizability of its generation process, the VSSM generates in two to eight time less time than the other methods, depending on the benchmark and the prompt.



(a) MNIST valid likelihood. (b) MNIST test statistics. (c) MNIST samples.



(d) CIFAR valid likelihood. (e) CIFAR test statistics. (f) CIFAR samples.

Figure 7.2: Samples, generation times and likelihoods of the Transformer, state space model and variational state space model for MNIST and CIFAR datasets. We report results over 5 runs of each model. Confidence intervals correspond to the minimum and maximum values observed. In 7.2a, 7.2d, we plot the median full and partial log-likelihood $\log p_\phi(x_{1:T})$ and $\log p_\phi(x_{C+1:T} \mid x_{1:C})$ on the validation set throughout training. In 7.2b, 7.2e, we report the average full and partial log-likelihood on the test set, along with mean execution times at generation, in both cases. In 7.2c, 7.2f, we report random qualitative examples for all models, for unconditioned sampling (first three rows) and conditioned on partial realizations (last three rows).

7.5 Conclusion

We introduce the VSSM, a dynamical VAE using SSMs as encoder and decoder. Compared to other architectures, our model is the first one that can generate in parallel while being recurrent, which allows generation to be resumed. Although tested on simple tasks, we show that it produces decent results in only a fraction of the time. The advantages of this architecture motivate further work to scale and improve performance on more challenging tasks such as language generation. It also motivates the investigation of its adaptation to more effective generative model such as hierarchical VAEs, discrete diffusion models, or score-based diffusion models.

7.A Mathematical Derivations

7.A.1 Learning Objective

Thanks to Jensen's inequality, we can show for dynamical VAEs of Subsection 7.2.1 that,

$$\log p_\phi(x_{1:T}) = \log \mathop{\mathbb{E}}_{p_\phi(z_{1:T})} p_\phi(x_{1:T}|z_{1:T}) \frac{q(z_{1:T}|x_{1:T})}{q(z_{1:T}|x_{1:T})}, \tag{7.11}$$

$$= \log \mathop{\mathbb{E}}_{q(z_{1:T}|x_{1:T})} \frac{p_\phi(x_{1:T}|z_{1:T}) p_\phi(z_{1:T})}{q(z_{1:T}|x_{1:T})}, \tag{7.12}$$

$$\geq \mathop{\mathbb{E}}_{q(z_{1:T}|x_{1:T})} \log \frac{p_\phi(x_{1:T}|z_{1:T}) p_\phi(z_{1:T})}{q(z_{1:T}|x_{1:T})}, \tag{7.13}$$

$$\geq \underbrace{\mathop{\mathbb{E}}_{q(z_{1:T}|x_{1:T})} \log p_\phi(x_{1:T}|z_{1:T}) - \mathrm{KL}(q(z_{1:T}|x_{1:T}) \parallel p_\phi(z_{1:T}))}_{\mathrm{ELBO}_\phi(x_{1:T})}. \tag{7.14}$$

Note that the ELBO becomes tight when the inference model $q(z_{1:T}|x_{1:T})$ corresponds to the true posterior distribution $p_\phi(z_{1:T}|x_{1:T})$. Indeed,

$$\mathrm{ELBO}_\phi(x_{1:T}) = \mathop{\mathbb{E}}_{q(z_{1:T}|x_{1:T})} \log \frac{p_\phi(z_{1:T}|x_{1:T}) p(x_{1:T})}{q(z_{1:T}|x_{1:T})}, \tag{7.15}$$

$$= \log p(x_{1:T}) - \mathrm{KL}(q(z_{1:T}|x_{1:T}) \parallel p_\phi(z_{1:T}|x_{1:T})), \tag{7.16}$$

and $\mathrm{KL}(q(z_{1:T}|x_{1:T}) \parallel p_\phi(z_{1:T}|x_{1:T})) = 0$ if and only if $q(z_{1:T}|x_{1:T}) = p_\phi(z_{1:T}|x_{1:T})$ almost everywhere. Hence, the dynamical VAE, composed of the prior $p_\phi(z_{1:T})$, generative model $p_\phi(x_{1:T}|z_{1:T})$ and inference model $q_\psi(z_{1:T}|x_{1:T})$ can be trained according to the objective function,

$$\max_{\phi,\psi} \mathop{\mathbb{E}}_{p(x_{1:T})} \left[\mathop{\mathbb{E}}_{q_\psi(z_{1:T}|x_{1:T})} [\log p_\phi(x_{1:T}|z_{1:T})] - \mathrm{KL}(q_\psi(z_{1:T}|x_{1:T}) \parallel p_\phi(z_{1:T})) \right]. \tag{7.17}$$

7.A.2 Approximate Partial Posterior

To learn the approximate partial posterior $q_\omega(z_{1:T}|x_{1:C})$ of the true partial posterior $p_\phi(z_{1:T}|x_{1:C})$ introduced in Subsection 7.3.2, we propose to exploit samples of the true partial posterior. Such samples are derived from samples $(x_{1:C}, x_{1:T})$, constructed from the dataset of sequences $x_{1:T}$ by taking random cuts $C \sim \mathcal{U}([0, T])$. Indeed, these allow us to draw samples $(x_{1:C}, z_{1:T})$ such that $z_{1:T} \sim p_\phi(z_{1:T}|x_{1:C})$, as suggested by the decomposition,

$$p_\phi(z_{1:T}|x_{1:C}) = \int_{x_{1:T}} p_\phi(z_{1:T}|x_{1:T}) p(x_{1:T}|x_{1:C}) \, \mathrm{d}x_{1:T}, \tag{7.18}$$

where $p_\phi(z_{1:T}|x_{1:T})$ is the true posterior of this VAE, which we approximate by the variational posterior $q_\psi(z_{1:T}|x_{1:T})$ during the VSSM training. The training objective for the approximate partial posterior $q_w(z_{1:T}|x_{1:C})$ is,

$$\arg\min_{\omega} \mathop{\mathbb{E}}_{p(x_{1:C})} \mathrm{KL}(p_\phi(z_{1:T}|x_{1:C}) \parallel q_\omega(z_{1:T}|x_{1:C})) \tag{7.19}$$

$$= \arg\min_{\omega} \mathop{\mathbb{E}}_{p(x_{1:C})} \mathop{\mathbb{E}}_{p_\phi(z_{1:T}|x_{1:C})} [\log p_\phi(z_{1:T}|x_{1:C}) - \log q_\omega(z_{1:T}|x_{1:C})] \tag{7.20}$$

$$= \arg\max_{\omega} \mathop{\mathbb{E}}_{p(x_{1:C})} \mathop{\mathbb{E}}_{p_\phi(z_{1:T}|x_{1:C})} [\log q_\omega(z_{1:T}|x_{1:C})] \tag{7.21}$$

$$= \arg\max_{\omega} \mathop{\mathbb{E}}_{p(x_{1:C})} \mathop{\mathbb{E}}_{p(x_{1:T}|x_{1:C})} [\mathop{\mathbb{E}}_{p_\phi(z_{1:T}|x_{1:T})} [\log q_\omega(z_{1:T}|x_{1:C})]] \tag{7.22}$$

$$\approx \arg\max_{\omega} \mathop{\mathbb{E}}_{p(x_{1:C})} \mathop{\mathbb{E}}_{p(x_{1:T}|x_{1:C})} [\mathop{\mathbb{E}}_{q_\psi(z_{1:T}|x_{1:T})} [\log q_\omega(z_{1:T}|x_{1:C})]]. \tag{7.23}$$

7.B Comparison of Autoregressive Generation

The VSSM sampling algorithm (Algorithm 7.1) can be compared to the RNN (Algorithm 7.2), SSM (Algorithm 7.3), and Transformer (Algorithm 7.4) algorithms. We also provide an algorithm for the chunk sampling method proposed in Subsection 7.3.2 in Algorithm 7.5.

Algorithm 7.1: Variational state space model sampling.

inputs: $x_{1:C}$ the prompt,
T the final length.

Let $\bar{v}_{1:T} = f_\omega^{\mathrm{par}}(k)$.
Sample $z_t \sim \mathcal{D}(z_t|\bar{v}_t),\ t = 1, \ldots, T$.
Let $w_{1:T} = f_\phi^{\mathrm{dec}}(z_{1:T})$.
Sample $x_t \sim \mathcal{P}(x_t|w_t),\ t = C+1, \ldots, T$.
return final sequence $x_{1:T}$.

Algorithm 7.2: Recurrent neural network sampling.

inputs: $x_{1:C}$ the prompt,
T the final length.

Let $h_0 = 0$.
for $t = 1, \ldots, C$ **do**
  Let $h_t = f_\phi(x_t, h_{t-1})$.
for $t = C+1, \ldots, T$ **do**
  Let $w_t = g_\phi(h_{t-1})$.
  Sample $x_t \sim \mathcal{D}(x_t|w_t)$.
  Let $h_t = f_\phi(x_t, h_{t-1})$.
return final sequence $x_{1:T}$.

7.C Additional Details

7.C.1 Training Hyperparameters

We train all three architectures (Transformer, SSM, VSSM) with comparable sizes for 200 epochs on the classical train set of the considered benchmarks (MNIST, CIFAR). To prevent overfitting, we use 10% of the train set for computing validation losses and likelihoods, as well as to select the final set of

Algorithm 7.3: State space model sampling.

inputs: $x_{1:C}$ the prompt,
T the final length.

Let $h_0 = 0$.
Let $h_C = f_\phi(x_{1:C}, h_0)$.
for $t = C+1, \ldots, T$ **do**
  Let $w_t = g_\phi(h_{t-1})$.
  Sample $x_t \sim \mathcal{D}(x_t|w_t)$.
  Let $h_t = f_\phi(x_t, h_{t-1})$.
return final sequence $x_{1:T}$.

Algorithm 7.4: Transformer sampling.

inputs: $x_{1:C}$ the prompt,
T the final length.

for $t = C+1, \ldots, T$ **do**
  Let $w_t = f_\phi(x_{1:t-1})$.
  Sample $x_t \sim \mathcal{D}(x_t|w_t)$.
return final sequence $x_{1:T}$.

Algorithm 7.5: Variational state space model chunk sampling.

inputs: $x_{1:C}$ the prompt,
T the final length,
$W \in [1, T-C]$ the chunk size.

Let $(\bar{v}_{1:C}, h_C^{\text{par}}) = f_\omega^{\text{par}}(x_{1:C})$.
Sample $z_t \sim \mathcal{D}(z_t|\bar{v}_t),\ t = 1, \ldots, C$.
Let $(w_{1:C}, h_C^{\text{dec}}) = f_\phi^{\text{dec}}(z_{1:C})$.
while $C \leq T$ **do**
  Let $(\bar{v}_{C+1:C+W}, h_{C+W}^{\text{par}}) = f_\omega^{\text{par}}(\varnothing_{C+1:C+W}, h_C^{\text{par}})$.
  Sample $z_t \sim \mathcal{D}(z_t|\bar{v}_t),\ t = C+1, \ldots, C+W$.
  Let $(w_{C+1:C+W}, h_{C+W}^{\text{dec}}) = f_\phi^{\text{dec}}(z_{C+1:C+W}, h_C^{\text{dec}})$.
  Sample $x_t \sim \mathcal{P}(x_t|w_t),\ t = C+1, \ldots, C+W$.
  Update $C = C + W$.
return final sequence $x_{1:T}$.

weights for evaluation on the test set and generating samples. All three architectures use 4 layers of dimension 1024 (with a state size of 16 for the SSM and VSSM, and with 8 heads of dimension $1024/8 = 128$ for the Transformer). We follow the attention block of GPT-2 for the Transformer, and the SSM block of Mamba for the SSM and VSSM architectures. The SSM and Transformer architectures output the mean of a Gaussian distribution with fixed variance ($\sigma = 0.1$), and are trained to maximize the log-likelihood. The VSSM generative model p_ϕ also outputs the mean of a Gaussian distribution with fixed variance ($\sigma = 0.1$), and is trained along with the posterior q_ψ to maximize the ELBO (7.14). Note that the temperature of the Gumbel softmax for computing $\nabla_\psi z_{1:T}$ was fixed to 1. The partial posterior q_ω is trained jointly with the encoder and decoder according to objective (7.23) and does not require to perform a subsequent training. All learning rates have been selected using a grid search in $(1 \times 10^{-2}, 5 \times 10^{-3}, 1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4})$.

7.C.2 Evaluation

We evaluate the likelihood by sampling $K = 100$ latent variables [Burda et al., 2015] from the posterior, and reweighting by the prior (resp. partial posterior), in order to measure the likelihood (resp. the partial likelihood),

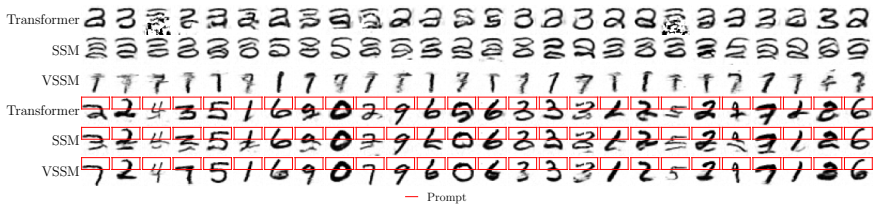
$$p_\phi(x_{1:T}) \approx \frac{1}{K} \sum_{k=1}^{K} p_\phi(x_{1:T}|z^k_{1:T}) \frac{p_\phi(z^k_{1:T})}{q_\psi(z^k_{1:T}|x_{1:T})}, \tag{7.24}$$

$$p_\phi(x_{C+1:T}|x_{1:C}) \approx \frac{1}{K} \sum_{k=1}^{K} p_\phi(x_{C+1:T}|z^k_{1:T}) \frac{q_\omega(z^k_{1:T}|x_{1:C})}{q_\psi(z^k_{1:T}|x_{1:T})}. \tag{7.25}$$

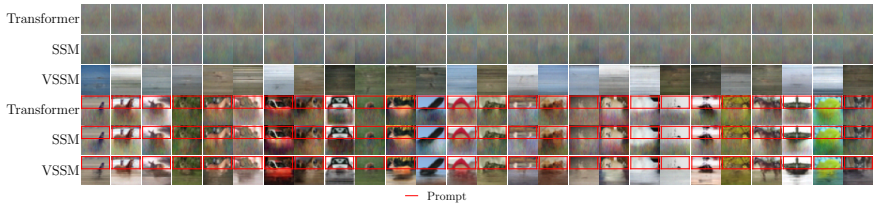
where $z^k_{1:T} \sim q_\psi(z^k_{1:T}|x_{1:T})$ in both cases. This expression is known to be a lower bound on the likelihood in expectation, and it tends towards the true likelihood as K grows to infinity.

7.D Additional Results

We report additional samples from all models in Figure 7.3a and Figure 7.3b, sampled randomly and using random prompts from the test set.



(a) MNIST samples.



(b) CIFAR samples.

Figure 7.3: Additional samples of the Transformer, state space model and variational state space model for MNIST and CIFAR datasets. We report random qualitative examples for all models, for unconditioned sampling (first three rows) and conditioned on partial realizations (last three rows).

Chapter 8

Just Looking at the Dice

In this chapter, we expose the promise of future works based on the variational state space model introduced in Chapter 7. By using a latent world model that decodes the latent variables of a history in parallel, and by conditioning the policy on the latent variables of that world model, we allow parallel imagination in model-based reinforcement learning.

8.1 Variational State Space Model

In this section, we recall the variational state space model (VSSM) [Lambrechts et al., 2024b], which is a structured variational autoencoder (VAE) [Kingma and Welling, 2014] based on state space models (SSM) [Gu et al., 2022b], where we select, given a target space $\mathcal{X}$,

- A discrete latent space of Z components of cardinality N each,

$$\mathcal{Z} = \{1, \dots, N\}^Z, \tag{8.1}$$

- A uniform prior distribution

$$p_\phi(z_{0:T}) = \prod_{t=1}^{T} p_\phi(z_t|z_{0:t-1}) = \prod_{t=1}^{T} p_\phi(z_t) = \prod_{t=1}^{T} \frac{1}{N^Z}, \tag{8.2}$$

- A generative distribution,

$$p_\phi(x_{0:T}|z_{0:T}) = \prod_{t=1}^{T} p_\phi(x_t|z_{0:t}) = \prod_{t=1}^{T} \mathcal{P}(x_t|f_\phi^{\text{dec}}(z_{0:t})), \tag{8.3}$$

  where $\mathcal{P}(x_t|w_t)$[1] is a distribution of parameters $w_t = f_\phi^{\text{dec}}(z_{0:t})$ outputted by a stacked SSM,

[1] Gaussian of mean w_t and fixed variance for continuous $\mathcal{X}$ or discrete distribution of probabilities w_t for discrete $\mathcal{X}$.

- A posterior distribution,

$$q_\psi(z_{0:T}|x_{0:T}) = \prod_{t=1}^{T} q_\psi(z_t|x_{0:t}) = \prod_{t=1}^{T} \mathcal{D}(z_t|f_\psi^{\text{enc}}(x_{0:t})), \tag{8.4}$$

  where $\mathcal{D}(z_t|v_t)$ is a discrete distribution of probabilities $v_t = f_\psi^{\text{enc}}(x_{0:t})$ outputted by a stacked SSM.

Because of the proposed conditional independences and thanks to the parallelizability of SSMs, the prior, posterior and generative models can be sampled in parallel along the time dimension [Lambrechts et al., 2024b].

8.2 Variational State Space World Model

In this section, we introduce an interesting usage of the VSSM in reinforcement learning (RL) for partially observable Markov decision processes (POMDP). Let us consider a POMDP $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, P, \gamma)$, with unobserved state $s_t \in \mathcal{S}$, actions $a_t \in \mathcal{A}$ and observation $o_t \in \mathcal{O}$, as defined in Chapter 2. Given the fact that the initial observation comes without reward nor previous action, let us conveniently define $r_{-1} = 0$ the null reward and $a_{-1} = 0$ the null action. Taking a sequence of t actions in the POMDP conditions its execution and provides the history $h_t = (o_0, a_0, \dots, o_t) \in \mathcal{H}$, where $\mathcal{H}$ is the set of histories of arbitrary length. A history-dependent world model aims at approximating the distribution $p(r, o'|h, a)$ for a given distribution $p(h, a)$ over the histories and actions. In practice, $p(h, a)$ is often chosen to the empirical distribution obtained when interacting with the POMDP using a policy that is jointly optimized with the world model.

Let us now introduce the variational state space world model (VSSWM) as a conditional VSSM with the following generative and posterior distributions,

- A generative distribution conditioned on the actions,

$$p_\phi(r_{-1:T-1}, o_{0:T}|z_{0:T}, a_{-1:T-1}) = \prod_{t=0}^{T} p_\phi(r_{t-1}, o_t|z_{0:t}, a_{-1:t-1}) \tag{8.5}$$

$$= \prod_{t=0}^{T} \mathcal{P}(r_{t-1}, o_t|f_\phi^{\text{dec}}(z_{0:t}, a_{-1:t-1})), \tag{8.6}$$

  where $\mathcal{P}(r_{t-1}, o_t|w_t)$ is a distribution of parameters $w_t = f_\phi^{\text{dec}}(z_{0:t}, a_{-1:t-1})$ outputted by a stacked SSM,

- A posterior distribution conditioned on the actions,

$$q_\psi(z_{0:T}|r_{-1:T-1}, o_{0:T}) = \prod_{t=0}^{T} q_\psi(z_t|r_{-1:t-1}, o_{0:t}) \tag{8.7}$$

$$= \prod_{t=0}^{T} \mathcal{D}(z_t|f_\psi^{\text{enc}}(r_{-1:t-1}, o_{0:t})), \tag{8.8}$$

  where $\mathcal{D}(z_t|v_t)$ is a discrete distribution of probabilities $v_t = f_\psi^{\text{enc}}(r_{-1:t-1}, o_{0:t})$ outputted by a stacked SSM.

This can simply be viewed as a standard VSSM, where $x_t = (r_{t-1}, o_t)$, but where the decoder is also conditioned on the previous action a_{t-1} together with the latent z_t at every time step.

8.3 Latent Policies and Parallel Imagination

Given a sequence of actions $a_{-1:T-1}$, the VSSWM is able to generate all rewards $r_{-1:T-1}$ and observations $o_{0:T}$, all in parallel, like a SSM-based world model [Samsami et al., 2024]. However, when training a policy in imagination [Hafner et al., 2020, Samsami et al., 2024], the sequence of actions in not known in advance. Indeed, we need the observation at time t for choosing the action at time t, which is in turn necessary for sampling the next observation at time $t+1$. This intrinsic sequentiality is characteristic of closed-loop control and RL.

Counterintuitively, we propose to avoid this sequentiality by simply coupling a latent policy implemented by an SSM with the parallelizable VSSWM. Let us define the deterministic history-dependent latent policy $\eta \colon \mathcal{H} \to \mathcal{A}$, such that,

$$a_t = f_\phi^{\text{act}}(z_{0:t}), \tag{8.9}$$

with f_ϕ^{act} a stacked SSM. The parallelizability of SSMs allows obtaining the complete sequence $a_{0:T}$ in parallel based on all latent variables $z_{0:T}$ sampled from the prior $p_\phi(z_{0:T})$. Since the latent variables encode all information about the realization of the interaction, the policy can indeed be conditioned on these sole variables. As a bonus, the rewards outputted by the world model are differentiable with respect to the parameters of the policy, at the condition that the actions are sampled in a differentiable way, for example with the reparametrization trick. It enables a direct reward maximization using stochastic gradient ascent for optimizing the policy without relying on standard RL techniques.

8.4 Conclusion

These ideas and their implications motivates the empirical investigation of such algorithms, that could unlock parallel imagination in model-based RL. Moreover, parallelizing the imagination could unlock new frontiers in terms of the imagination horizon, which could help in learning policies that have a long planning horizon. Similarly, a more efficient imagination could unlock new frontiers in meta-learning when we want to train the policy on many environments in parallel. While not developed here, it would also be interesting to adapt the autoregressive version of the VSSM, so as to be able to resume imagination without reprocessing the complete history. This project probably also present some limitations, since the results previously obtained with the VSSM were limited, and since this approach puts a heavy workload on the policy, which needs to internally decode the latent variables of the world model.

Conclusion

Chapter 9

A Matter of Abstractions

By considering perception as the inception of decision making, this thesis has explored the intricate relationships between abstractions, memory, predictions and decisions, when learning to act optimally. Rooting this research project in the optimal control theory allowed a deep understanding of the desired solution, which motivated the analyses and contributions of this thesis. We notably demonstrated the importance of explicitly considering the structure of the solution, moving beyond the simplistic view of the history Markov decision process. We highlighted the similarity between the abstractions learned by such uninformed approaches and the sufficient representations prescribed by theory. We also explored the possibility of fostering abstractions of the perception that are as predictive as these sufficient representations. More broadly, our research has argued in favor of entangling the task of processing of the past and the task of planning for the future, but also in favor of their separate consideration through different objectives, acknowledging their different nature. These conclusions were developed in three thematic parts that studied reinforcement learning (RL) in partially observable Markov decision processes (POMDP).

The first part of the thesis studied the role of memory when learning to act optimally. We showed empirically that recurrent neural networks, when trained to act optimally in a POMDP, implicitly learn representations that are related to the belief, that is the posterior distribution over the states given the history. The similarity of these learned representations with the belief also motivated representations objectives that foster the memory to embed such information, which was explored in the second part of this thesis. We also demonstrated the ability of history-dependent RL in POMDP to learn to discard the belief of irrelevant state variables from their representations and to only focus on relevant variables for optimal control. Furthermore, we also demonstrated the decisive importance of memory for learning efficiently. More precisely, we introduced an initialization procedure to maximize the multistability of a recurrent neural network. We showed empirically that it could endow any recurrent neural network with long-lasting memorization abilities, improving its ability to learn in the presence of long time dependencies. In summary, these findings highlighted the importance of learning good representations of the past, and motivated to explicitly learn such representations. It also highlighted the importance of having

representations that encode a lot of information at the initial stages of learning.

The second part of the thesis addressed the challenge of explicitly learning sufficient representations of the history, by leveraging eventual additional information about the state to learn from. We started by relaxing the usual asymmetric learning formalization, by allowing any information about the state to be considered for improving the learning process. Then, noticing that model-based methods embed representation objectives by learning statistics that are predictive of future observations, we extended these methods to learn statistics that are predictive of all available state information. It resulted in the informed Dreamer algorithm, allowing to predict information about the state, instead of explicitly predicting observations, while still allowing to learn latent policies in imagination with the informed world model. By leveraging both the additional information and the efficiency of model-based RL, it touched on the state-of-the-art for learning policies in POMDPs. Then, convinced by the empirical merits of the asymmetric RL approach, we questioned its theoretical foundations. Existing methods were theoretically sound in the sense that they provide optimal history-dependent policies after convergence, but they still lacked a justification for their potential benefits. By analyzing a simple version of the asymmetric actor-critic algorithm, we demonstrated a possible reason for the effectiveness of such an algorithm, compared to its symmetric counterpart. The finite-time convergence analysis of the asymmetric algorithm showed that it eliminates an error arising from insufficient history representations when performing temporal difference learning, compared to the symmetric algorithm.

The third part of the thesis explored future avenues where the interplays between representations, memory, predictions and decisions are completely entangled. We first introduced the variational state space model, a novel sequence modeling architecture that enables parallel generation while retaining the benefits of recurrent models. This model was shown to compete with classical autoregressive models such as transformers and state space models on toy benchmarks, while generating samples in a fraction of the time, thanks to its parallelizability along the time dimension. We then proposed its use as a world model in model-based RL, by presenting the theoretical advantages of such a method, but without studying it empirically. By conditioning parallelizable policies on latent representations of past and future realizations of such world models, this line of research could eliminate the sequential nature of the closed-loop control paradigm of RL. It goes in the direction motivated by this thesis, which is to entangle the processing of the past and the prediction of the future, by sharing representations between the world model and the policy, while still keeping distinct learning objectives to exploit the structure of the solution.

As a conclusion, this thesis established a clear motivation for the particular attention that should be given to the problem of abstracting the past, for then forming its decision. For this purpose, this thesis has studied, and improved a variety of RL methods, humbly improving the intelligent behaviors that can be learned from interaction with a POMDP. This thesis also motivates many unexplored future works, such as (i) learning sufficient statistics in an asymmetric setting using contrastive learning or mutual information maximization, (ii) designing recurrent neural networks that have nonlinear and multistable dynamics while being parallelizable along the time dimension, (iii) improving asymmetric

world models with contrastive representation learning to design minimal representations of the additional information that are relevant to the control task, (iv) empirically studying the aliasing in symmetric and asymmetric recurrent actor-critic algorithms, and (v) improving variational state space models using more expressive generative models and considering their usage as a world model, to quote only one idea from each paper.

As a closing remark, I want to highlight that we did not considered the ultimate decision making problem that RL aims at solving, which is the problem of acting optimally across the distribution of all problems that can be encountered. This generalization problem, which consists in learning to act for any situation, can elegantly be formalized as a POMDP. Intuitively, the problem of acting optimally over a distribution of unknown decision processes indeed amounts to inferring the current control problem from perception, for then adjusting its decisions and acting optimally in the future. More formally, the parameters characterizing the actual process we are currently tasked with controlling can be seen as a state variable, whose belief can be inferred from perception. It should moreover be noted that the solution of this particular POMDP probably has an interesting structure, where the state variables that describe the current control problem have a particular dynamic. We view as a wonderful future work the application of the findings of this thesis, and more broadly the findings of the literature on partial observability, to this generalization problem.

Bibliography

Bibliography

Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. On the Theory of Policy Gradient Methods: Optimality, Approximation, and Distribution Shift. *Journal of Machine Learning Research*, 2021.

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Ma teusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving Rubik's Cube with a Robot Hand. *arXiv:1910.07113*, 2019.

Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc-Alexandre Côté, and R Devon Hjelm. Unsupervised State Representation Learning in Atari. *Advances in Neural Information Processing Systems*, 2019.

Karl Johan Åström. Optimal Control of Markov Processes with Incomplete State Information. *Journal of Mathematical Analysis and Applications*, 1965.

Karl Johan Åström and Peter Eykhoff. System Identification - A Survey. *Automatica*, 1971.

Raphaël Avalos, Florent Delgrange, Ann Nowe, Guillermo Perez, and Diederik M Roijers. The Wasserstein Believer: Learning Belief Updates for Partially Observable Environments through Reliable Latent Space Models. *The Twelfth International Conference on Learning Representations*, 2024.

Andrea Baisero and Christopher Amato. Unbiased Asymmetric Reinforcement Learning under Partial Observability. *International Conference on Autonomous Agents and Multiagent Systems*, 2022.

Andrea Baisero, Brett Daley, and Christopher Amato. Asymmetric DQN for Partially Observable Reinforcement Learning. *Conference on Uncertainty in Artificial Intelligence*, 2022.

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

Stefan Banach. Sur les Opérations dans les Ensembles Abstraits et leur Application aux Équations Intégrales. *Fundamenta Mathematicae*, 1922.

Thomas Bayes. An Essay Towards Solving a Problem in the Doctrine of Chances. *Philosophical Transactions of the Royal Society of London*, 1763.

Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeshwar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and Devon Hjelm. Mutual Information Neural Estimation. *International Conference on Machine Learning*, 2018.

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

Richard Bellman. On the Theory of Dynamic Programming. *National Academy of Sciences of the United States of America*, 1952.

Richard Bellman. A Markovian Decision Process. *Indiana University Mathematics Journal*, 1957.

Yoshua Bengio, Paolo Frasconi, and Patrice Simard. The Problem of Learning Long-Term Dependencies in Recurrent Networks. *IEEE International Conference on Neural Networks*, 1993.

José M Bernardo and Adrian FM Smith. *Bayesian Theory*. John Wiley & Sons, 2009.

Dimitri Bertsekas. *Dynamic Programming and Optimal Control: Volume I*. Athena Scientific, 2012.

Guy E Blelloch. Prefix Sums and Their Applications. *School of Computer Science, Carnegie Mellon University Pittsburgh, PA, USA*, 1990.

Adrien Bolland, Gaspard Lambrechts, and Damien Ernst. Behind the Myth of Exploration in Policy Gradients. *arXiv:2402.00162*, 2024a.

Adrien Bolland, Gaspard Lambrechts, and Damien Ernst. Off-Policy Maximum Entropy RL with Future State and Action Visitation Measures. *arXiv: 2412.06655*, 2024b.

Lars Buesing, Théophane Weber, Sébastien Racanière, S. M. Ali Eslami, Danilo Jimenez Rezende, David P. Reichert, Fabio Viola, Frederic Besse, Karol Gregor, Demis Hassabis, and Daan Wierstra. Learning and Querying Fast Generative Models for Reinforcement Learning. *arXiv:1802.03006*, 2018.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance Weighted Autoencoders. *arXiv:1509.00519*, 2015.

Yang Cai, Xiangyu Liu, Argyris Oikonomou, and Kaiqing Zhang. Provable Partially Observable Reinforcement Learning with Privileged Information. *Advances in Neural Information Processing Systems*, 2024.

Semih Cayci and Atilla Eryilmaz. Convergence of Gradient Descent for Recurrent Neural Networks: A Nonasymptotic Analysis. *arXiv:2402.12241*, 2024a.

Semih Cayci and Atilla Eryilmaz. Recurrent Natural Policy Gradient for POMDPs. *ICML Workshop on the Foundations of Reinforcement Learning and Control*, 2024b.

Semih Cayci, Niao He, and R Srikant. Finite-Time Analysis of Natural Actor-Critic for POMDPs. *SIAM Journal on Mathematics of Data Science*, 2024.

Andrea Ceni, Peter Ashwin, and Lorenzo Livi. Interpreting Recurrent Neural Networks Behaviour via Excitable Network Attractors. *Cognitive Computation*, 2020.

Jinmiao Chen and Narendra S Chaudhari. Segmented-Memory Recurrent Neural Networks. *IEEE Transactions on Neural Networks*, 2009.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. *arXiv:1409.1259*, 2014a.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *Conference on Empirical Methods in Natural Language Processing*, 2014b.

Sanjiban Choudhury, Mohak Bhardwaj, Sankalp Arora, Ashish Kapoor, Gireeja Ranade, Sebastian Scherer, and Debadeepta Dey. Data-Driven Planning via Imitation Learning. *The International Journal of Robotics Research*, 2018.

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

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv:1412.3555*, 2014.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A Recurrent Latent Variable Model for Sequential Data. *Advances in Neural Information Processing Systems*, 2015.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical Multiscale Recurrent Neural Networks. *arXiv:1609.01704*, 2017.

Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan D. Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, J-M. Moret, Seb Noury, Federico Pesamosca, D. Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, B. Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin A. Riedmiller. Magnetic Control of Tokamak Plasmas through Deep Reinforcement Learning. *Nature*, 2022.

Shi Dong, Benjamin Van Roy, and Zhengyuan Zhou. Simple Agent, Complex Environment: Efficient Reinforcement Learning with Agent States. *Journal of Machine Learning Research*, 2022.

Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic Evaluation of Certain Markov Process Expectations for Large Time. *Communications on Pure and Applied Mathematics*, 1975.

Kenji Doya. Bifurcations of Recurrent Neural Networks in Gradient Descent Learning. *IEEE Transactions on Neural Networks*, 1993.

Le Fang, Tao Zeng, Chaochun Liu, Liefeng Bo, Wen Dong, and Changyou Chen. Transformer-Based Conditional Variational Autoencoder for Controllable Story Generation. *arXiv:2101.00828*, 2021.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual Multi-Agent Policy Gradients. *AAAI Conference on Artificial Intelligence*, 2018.

Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P Adams, and Sergey Levine. Why Generalization in RL is Difficult: Epistemic POMDPs and Implicit Partial Observability. *Advances in Neural Information Processing Systems*, 2021.

Laurent Girin, Simon Leglaive, Xiaoyu Bie, Julien Diard, Thomas Hueber, and Xavier Alameda-Pineda. Dynamical Variational Autoencoders: A Comprehensive Review. *Foundations and Trends in Machine Learning*, 2021.

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. DiffuSeq: Sequence to Sequence Text Generation with Diffusion Models. *International Conference on Learning Representations*, 2023.

Alex Graves. Generating Sequences with Recurrent Neural Networks. *arXiv: 1308.0850*, 2013.

Karol Gregor, Danilo Jimenez Rezende, Frederic Besse, Yan Wu, Hamza Merzic, and Aaron van den Oord. Shaping Belief States with Generative Environment Models for RL. *Advances in Neural Information Processing Systems*, 2019.

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

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

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

Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Bernardo A Pires, and Ré Munosmi. Neural Predictive Belief Representations. *arXiv: 1811.06407*, 2018.

Zhaohan Daniel Guo, Bernardo Avila Pires, Bilal Piot, Jean-Bastien Grill, Florent Altché, Rémi Munos, and Mohammad Gheshlaghi Azar. Bootstrap Latent-Predictive Representations for Multitask Reinforcement Learning. *International Conference on Machine Learning*, 2020.

Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal State Spaces are as Effective as Structured State Spaces. *Advances in Neural Information Processing Systems*, 2022.

David Ha and Jürgen Schmidhuber. Recurrent World Models Facilitate Policy Evolution. *Advances in Neural Information Processing Systems*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *International Conference on Machine Learning*, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning Latent Dynamics for Planning from Pixels. *International Conference on Machine Learning*, 2019.

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

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering Atari with Discrete World Models. *International Conference on Learning Representations*, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering Diverse Domains through World Models. *arXiv:2301.04104*, 2023.

Dongqi Han, Kenji Doya, and Jun Tani. Variational Recurrent Models for Solving Partially Observable Control Tasks. *Internal Conference on Learning Representations*, 2019.

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

Nicolas Heess, Jonathan J Hunt, Timothy P Lillicrap, and David Silver. Memory-Based Control with Recurrent Neural Networks. *arXiv:1512.04455*, 2015.

Jay A Hennig, Sandra A Romero Pinto, Takahiro Yamaguchi, Scott W Linderman, Naoshige Uchida, and Samuel J Gershman. Emergence of Belief-Like Representations through Reinforcement Learning. *PLoS Computational Biology*, 2023.

Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining Improvements in Deep Reinforcement Learning. *AAAI Conference on Artificial Intelligence*, 2018.

Salah Hihi and Yoshua Bengio. Hierarchical Recurrent Neural Networks for Long-Term Dependencies. *Advances in Neural Information Processing Systems*, 1995.

YC Ho and RCKA Lee. A Bayesian Approach to Problems in Stochastic Estimation and Control. *IEEE Transactions on Automatic Control*, 1964.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 1997.

Yitian Hong, Yaochu Jin, and Yang Tang. Rethinking Individual Global Max in Cooperative Multi-Agent Reinforcement Learning. *Advances in Neural Information Processing Systems*, 2022.

Dino Ienco, Roberto Interdonato, and Raffaele Gaetano. Supervised Level-Wise Pretraining for Recurrent Neural Network Initialization in Multi-Class Classification. *arXiv:1911.01071*, 2019.

Maximilian Igl, Luisa Zintgraf, Tuan Anh Le, Frank Wood, and Shimon Whiteson. Deep Variational Reinforcement Learning for POMDPs. *International Conference on Machine Learning*, 2018.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical Reparameterization with Gumbel-Softmax. *International Conference on Learning Representations*, 2017.

Junyan Jiang, Gus G Xia, Dave B Carlton, Chris N Anderson, and Ryan H Miyakawa. Transformer VAE: A Hierarchical Model for Structure-Aware and Interpretable Music Representation Learning. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2020.

José Maria P Menezes Jr and Guilherme A Barreto. Long-Term Time Series Prediction with the NARX Network: An Empirical Evaluation. *Neurocomputing*, 2008.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 1998.

Sham Kakade and John Langford. Approximately Optimal Approximate Reinforcement Learning. *International Conference on Machine Learning*, 2002.

Sham M Kakade. A Natural Policy Gradient. *Advances in Neural Information Processing Systems*, 2001.

Peter Karkus, David Hsu, and Wee Sun Lee. QMDP-net: Deep learning for planning under partial observability. *Advances in Neural Information Processing Systems*, 2017.

Garrett E Katz and James A Reggia. Using Directional Fibers to Locate Fixed Points of Recurrent Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2017.

Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Champion-Level Drone Racing using Deep Reinforcement Learning. *Nature*, 2023.

Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. *International Conference on Learning Representations*, 2014.

Jan Koutnik, Klaus Greff, Faustino Gomez, and Jürgen Schmidhuber. A Clockwork RNN. *International Conference on Machine Learning*, 2014.

Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating Mutual Information. *Physical Review*, 2004.

Gaspard Lambrechts, Adrien Bolland, and Damien Ernst. Recurrent Networks, Hidden States and Beliefs in Partially Observable Environments. *Transactions on Machine Learning Research*, 2022.

Gaspard Lambrechts, Florent De Geeter, Nicolas Vecoven, Damien Ernst, and Guillaume Drion. Warming Up Recurrent Neural Networks to Maximise Reachable Multistability Greatly Improves Learning. *Neural Networks*, 2023.

Gaspard Lambrechts, Adrien Bolland, and Damien Ernst. Informed POMDP: Leveraging Additional Information in Model-Based RL. *Reinforcement Learning Journal*, 2024a.

Gaspard Lambrechts, Yann Claes, Pierre Geurts, and Damien Ernst. Parallelizing Autoregressive Generation with Variational State Space Models. *ICML Workshop on the Next Generation of Sequence Modeling Architectures*, 2024b.

Gaspard Lambrechts, Damien Ernst, and Aditya Mahajan. A Theoretical Justification for Asymmetric Actor-Critic Algorithms. *International Conference on Machine Learning*, 2025.

Alex X Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic Latent Actor-Critic: Deep Reinforcement Learning with a Latent Variable Model. *Advances in Neural Information Processing Systems*, 2020.

Pascal Leroy, Damien Ernst, Pierre Geurts, Gilles Louppe, Jonathan Pisane, and Matthia Sabatelli. QVMix and QVMix-Max: Extending the Deep Quality-Value Family of Algorithms to Cooperative Multi-Agent Reinforcement Learning. *AAAI Workshop on Reinforcement Learning in Games*, 2021.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-End Training of Deep Visuomotor Policies. *Journal of Machine Learning Research*, 2015.

Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. Robust Multi-Agent Reinforcement Learning via Minimax Deep Deterministic Policy Gradient. *AAAI Conference on Artificial Intelligence*, 2019.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous Control with Deep Reinforcement Learning. *arXiv:1509.02971*, 2015.

Long-Ji Lin and Tom M Mitchell. Memory Approaches to Reinforcement Learning in Non-Markovian Domains, 1992.

Tsungnan Lin, Bill G Horne, Peter Tino, and C Lee Giles. Learning Long-Term Dependencies in NARX Recurrent Neural Networks. *IEEE Transactions on Neural Networks*, 1996.

Michael Littman and Richard S Sutton. Predictive Representations of State. *Advances in Neural Information Processing Systems*, 2001.

Michael L Littman, Anthony R Cassandra, and Leslie Pack Kaelbling. Learning Policies for Partially Observable Environments: Scaling Up. *International Conference on Machine Learning*, 1995.

Danyang Liu and Gongshen Liu. A Transformer-Based Variational Autoencoder for Sentence Generation. *2019 International Joint Conference on Neural Networks*, 2019.

Arthur Louette, Gaspard Lambrechts, Damien Ernst, Eric Pirard, and Godefroid Dislaire. Reinforcement Learning to Improve Delta Robot Throws for Sorting Scrap Metal. *arXiv:2406.13453*, 2024.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Advances in Neural Information Processing Systems*, 2017.

Xueguang Lyu and Yuchen Xiao. A Deeper Understanding of State-Based Critics in Multi-Agent Reinforcement Learning. *AAAI Conference on Artificial Intelligence*, 2022.

Xiao Ma, Peter Karkus, David Hsu, Wee Sun Lee, and Nan Ye. Discriminative Particle Filter Reinforcement Learning for Complex Partial Observations. *International Conference on Learning Representations*, 2020.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables. *International Conference on Learning Representations*, 2016.

Niru Maheswaranathan, Alex Williams, Matthew Golub, Surya Ganguli, and David Sussillo. Reverse Engineering Recurrent Networks for Sentiment Classification Reveals Line Attractor Dynamics. *Advances in Neural Information Processing Systems*, 2019.

John McCarthy. What is Artificial Intelligence, 1998.

Tomas Mikolov, Armand Joulin, Sumit Chopra, Michael Mathieu, and Marc'Aurelio Ranzato. Learning Longer Memory in Recurrent Neural Networks. *International Conference on Learning Representations*, 2015.

Vladimir Mikulik, Grégoire Delétang, Tom McGrath, Tim Genewein, Miljan Martic, Shane Legg, and Pedro Ortega. Meta-Trained Agents Implement Bayes-Optimal Agents. *Advances in Neural Information Processing Systems*, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Kirkeby Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-Level Control through Deep Reinforcement Learning. *Nature*, 2015.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. *International Conference on Machine Learning*, 2016.

Steven Morad, Ryan Kortvelesy, Matteo Bettini, Stephan Liwicki, and Amanda Prorok. POPGym: Benchmarking Partially Observable Reinforcement Learning. *International Conference on Learning Representations*, 2023.

Hai Nguyen, Brett Daley, Xinchao Song, Christopher Amato, and Robert Platt. Belief-Grounded Networks for Accelerated Robot Learning under Partial Observability. *Conference on Robot Learning*, 2021.

Tianwei Ni, Benjamin Eysenbach, Erfan SeyedSalehi, Michel Ma, Clement Gehring, Aditya Mahajan, and Pierre-Luc Bacon. Bridging State and History Representations: Understanding Self-Predictive RL. *International Conference on Learning Representations*, 2024.

Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and Approximate Q-Value Functions for Decentralized POMDPs. *Journal of Artificial Intelligence Research*, 2008.

Bun Theang Ong, Komei Sugiura, and Koji Zettsu. Dynamic Pre-Training of Deep Recurrent Neural Networks for Predicting Environmental Monitoring Data. *IEEE International Conference on Big Data*, 2014.

Antonio Orvieto, Soham De, Caglar Gulcehre, Razvan Pascanu, and Samuel L Smith. On the Universality of Linear Recurrences Followed by Nonlinear Projections. *arXiv:2307.11888*, 2023.

Luca Pasa and Alessandro Sperduti. Pre-Training of Recurrent Neural Networks via Linear Autoencoders. *Advances in Neural Information Processing Systems*, 2014.

Luca Pasa, Alberto Testolin, and Alessandro Sperduti. Neural Networks for Sequential Data: a Pre-Training Approach Based on Hidden Markov Models. *Neurocomputing*, 2015.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the Difficulty of Training Recurrent Neural Networks. *International Conference on Machine Learning*, 2013.

Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric Actor Critic for Image-Based Robot Learning. *Robotics: Science and Systems*, 2018.

Josep M Porta, Nikos Vlassis, Matthijs TJ Spaan, and Pascal Poupart. Point-Based Value Iteration for Continuous POMDPs. *Journal of Machine Learning Research*, 2006.

Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. *OpenAI Blog*, 2019.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *International Conference on Machine Learning*, 2018.

Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted QMix: Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *Advances in Neural Information Processing Systems*, 2020.

Alaa Sagheer and Mostafa Kotb. Unsupervised Pre-Training of a Deep LSTM-Based Stacked Autoencoder for Multivariate Time Series Forecasting Problems. *Scientific Reports*, 2019.

Mohammad Reza Samsami, Artem Zholus, Janarthanan Rajendran, and Sarath Chandar. Mastering Memory Tasks with World Models. *International Conference on Learning Representations*, 2024.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, L. Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *Nature*, 2020.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust Region Policy Optimization. *International Conference on Machine Learning*, 2015.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

Yang Shao, Quan Kong, Tadayuki Matsumura, Taiki Fuji, Kiyoto Ito, and Hiroyuki Mizuno. Mask Atari for Deep reinforcement Learning as POMDP Benchmarks. *arXiv:2203.16777*, 2022.

Amit Sinha and Aditya Mahajan. Asymmetric Actor-Critic with Approximate Information State. *IEEE Conference on Decision and Control*, 2023.

Amit Sinha and Aditya Mahajan. Agent-State Based Policies in POMDPs: Beyond Belief-State MDPs. *arXiv:2409.15703*, 2024.

Richard D Smallwood and Edward J Sondik. The Optimal Control of Partially Observable Markov Processes over a Finite Horizon. *Operations Research*, 1973.

Jimmy TH Smith, Andrew Warrington, and Scott Linderman. Simplified State Space Layers for Sequence Modeling. *The Eleventh International Conference on Learning Representations*, 2023.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. *International Conference on Machine Learning*, 2019.

Edward J Sondik. The Optimal Control of Partially Observable Markov Processes over the Infinite Horizon: Discounted Costs. *Operations Research*, 1978.

Matthijs TJ Spaan. Partially Observable Markov Decision Processes. In *Reinforcement Learning: State-of-the-Art*. Springer, 2012.

Charlotte T Striebel. Sufficient Statistics in the Optimum Control of Stochastic Systems. *Journal of Mathematical Analysis and Applications*, 1965.

Jayakumar Subramanian, Amit Sinha, Raihan Seraj, and Aditya Mahajan. Approximate Information State for Approximate Planning and Reinforcement Learning in Partially Observed Systems. *Journal of Machine Learning Research*, 2022.

David Sussillo and Omri Barak. Opening the Black Box: Low-Dimensional Dynamics in High-Dimensional Recurrent Neural Networks. *Neural Computation*, 2013.

Richard S Sutton. Dyna, an Integrated Architecture for Learning, Planning, and Reacting. *ACM Sigart Bulletin*, 1991.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. *Advances in Neural Information Processing Systems*, 1999.

Corentin Tallec and Yann Ollivier. Can Recurrent Neural Networks Warp Time? *arXiv:1804.11188*, 2018.

Zhiyuan Tang, Dong Wang, and Zhiyong Zhang. Recurrent Neural Network Training with Dark Knowledge Transfer. *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2016.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. Deepmind Control Suite. *arXiv:1801.00690*, 2018.

Sebastian Thrun. *Probabilistic Robotics*. The MIT Press, 2002.

Trieu Trinh, Andrew Dai, Thang Luong, and Quoc Le. Learning Longer-Term Dependencies in RNNs with Auxiliary Losses. *International Conference on Machine Learning*, 2018.

Miguel Vasco, Takuma Seno, Kenta Kawamoto, Kaushik Subramanian, Peter R Wurman, and Peter Stone. A Super-Human Vision-Based Reinforcement Learning Agent for Autonomous Racing in Gran Turismo. *Reinforcement Learning Journal*, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Advances in Neural Information Processing Systems*, 2017.

Nicolas Vecoven, Damien Ernst, and Guillaume Drion. A Bio-Inspired Bistable Recurrent Cell allows for Long-Lasting Memory. *PloS One*, 2021.

Andrew Wang, Andrew C Li, Toryn Q Klassen, Rodrigo Toro Icarte, and Sheila A McIlraith. Learning Belief Representations for Partially Observable Deep RL. *International Conference on Machine Learning*, 2023.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. QPLEX: Duplex Dueling Multi-Agent Q-Learning. *International Conference on Learning Representations*, 2021.

Rose E Wang, Michael Everett, and Jonathan P How. R-MADDPG for Partially Observable Environments and Limited Communication. *arXiv:2002.06684*, 2020.

Andrew Warrington, Jonathan W Lavington, Adam Scibior, Mark Schmidt, and Frank Wood. Robust Asymmetric Learning in POMDPs. *International Conference on Machine Learning*, 2021.

Chris Watkins. *Learning from Delayed Rewards*. PhD thesis, Cambridge University, Cambridge, England, 1989.

Paul J Werbos. Backpropagation Through Time: What It Does and How to Do It. *Proceedings of the IEEE*, 1990.

Jos Van Der Westhuizen and Joan Lasenby. The Unreasonable Effectiveness of the Forget Gate. *arXiv:1804.04849*, 2018.

Daan Wierstra, Alexander Förster, Jan Peters, and Jürgen Schmidhuber. Solving Deep Memory POMDPs with Recurrent Policy Gradients. *International Conference on Artificial Neural Networks*, 2007.

Ronald J Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 2004.

Ronald J Williams and David Zipser. Gradient-Based Learning Algorithms for Recurrent Networks and Their Computational Complexity. In *Backpropagation*. Psychology Press, 2013.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep Sets. *Advances in Neural Information Processing Systems*, 2017.

Marvin Zhang, Zoe McCarthy, Chelsea Finn, Sergey Levine, and Pieter Abbeel. Learning Deep Neural Network Policies with Continuous Memory States. *IEEE International Conference on Robotics and Automation*, 2016.

Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine. SOLAR: Deep Structured Representations for Model-Based Reinforcement Learning. *International Conference on Machine Learning*, 2019.

Guo-Bing Zhou, Jianxin Wu, Chen-Lin Zhang, and Zhi-Hua Zhou. Minimal Gated Unit for Recurrent Neural Networks. *International Journal of Automation and Computing*, 2016.

Linqi Zhou, Michael Poli, Winnie Xu, Stefano Massaroli, and Stefano Ermon. Deep Latent State Space Models for Time-Series Generation. *International Conference on Machine Learning*, 2023.

Pengfei Zhu, Xin Li, Pascal Poupart, and Guanghui Miao. On Improving Deep Reinforcement Learning for POMDPs. *arXiv:1704.07978*, 2017.

Appendices

Appendix A

Belief Recurrence

In this appendix, we demonstrate the recurrence of the belief update [Ho and Lee, 1964]. More formally, we consider a partially observable Markov decision process (POMDP) $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, P, \gamma)$ with discrete state space $\mathcal{S}$, discrete action space $\mathcal{A}$ and discrete observation space $\mathcal{O}$. Let $b_{t+1} \in \mathcal{B}$ be the belief of a history $h_{t+1} \in \mathcal{H}$. The belief $b_{t+1} = f(h_{t+1})$ can be written as,

$$
\begin{aligned}
b_{t+1}(s_{t+1}) &= \Pr(s_{t+1}|h_{t+1}) && \text{(A.1)}\\
&= \Pr(s_{t+1}|h_t, a_t, o_{t+1}) && \text{(A.2)}\\
&= \frac{\Pr(s_{t+1}, o_{t+1}|h_t, a_t)}{\Pr(o_{t+1}|h_t, a_t)} && \text{(A.3)}\\
&= \frac{\Pr(s_{t+1}, o_{t+1}|h_t, a_t)}{\sum_{s_{t+1}\in\mathcal{S}} \Pr(s_{t+1}, o_{t+1}|h_t, a_t)} && \text{(A.4)}\\
&= \frac{\Pr(o_{t+1}|s_{t+1}, h_t, a_t)\Pr(s_{t+1}|h_t, a_t)}{\sum_{s_{t+1}\in\mathcal{S}} \Pr(o_{t+1}|s_{t+1}, h_t, a_t)\Pr(s_{t+1}|h_t, a_t)} && \text{(A.5)}\\
&= \frac{\Pr(o_{t+1}|s_{t+1})\Pr(s_{t+1}|h_t, a_t)}{\sum_{s_{t+1}\in\mathcal{S}} \Pr(o_{t+1}|s_{t+1})\Pr(s_{t+1}|h_t, a_t)} && \text{(A.6)}\\
&= \frac{O(o_{t+1}|s_{t+1})\Pr(s_{t+1}|h_t, a_t)}{\sum_{s_{t+1}\in\mathcal{S}} O(o_{t+1}|s_{t+1})\Pr(s_{t+1}|h_t, a_t)}. && \text{(A.7)}
\end{aligned}
$$

Let us now focus on $\Pr(s_{t+1}|h_t, a_t)$, which can be developed as follows,

$$
\begin{aligned}
\Pr(s_{t+1}|h_t, a_t) &= \sum_{s_t\in\mathcal{S}} \Pr(s_{t+1}, s_t|h_t, a_t) && \text{(A.8)}\\
&= \sum_{s_t\in\mathcal{S}} \Pr(s_{t+1}|s_t, h_t, a_t)\Pr(s_t|h_t, a_t) && \text{(A.9)}\\
&= \sum_{s_t\in\mathcal{S}} \Pr(s_{t+1}|s_t, a_t)\Pr(s_t|h_t) && \text{(A.10)}\\
&= \sum_{s_t\in\mathcal{S}} T(s_{t+1}|s_t, a_t) b_t(s_t), && \text{(A.11)}
\end{aligned}
$$

with $b_t = f(h_t)$ and where the penultimate line holds at the condition that the actions are conditionally independent of the state given the history, that is

$\Pr(s_t|h_t, a_t) = \Pr(s_t|h_t)$, which is the case when considering history-dependent policies $\pi\colon \mathcal{H} \to \Delta(\mathcal{A})$. By substituting equation (A.11) in equation (A.7), we obtain,

$$b_{t+1}(s_{t+1}) = \frac{O(o_{t+1}|s_{t+1}) \sum_{s_t \in \mathcal{S}} T(s_{t+1}|s_t, a_t) b_t(s_t)}{\sum_{s_{t+1} \in \mathcal{S}} O(o_{t+1}|s_{t+1}) \sum_{s_t \in \mathcal{S}} T(s_{t+1}|s_t, a_t) b_t(s_t)}. \tag{A.12}$$

Appendix B

Belief Markov Decision Processes

In this appendix, we demonstrate how partially observable Markov decision processes (POMDP) can be reduced to Markov decision processes (MDP) using the recurrent belief update [Åström, 1965]. More formally, we derive the equation of the equivalent belief MDP $\mathcal{M}' = (\mathcal{S}', \mathcal{A}, T', R', P', \gamma)$ for a given POMDP $\mathcal{P} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, O, P, \gamma)$. The belief MDP state is $S' = \mathcal{B}$, where $\mathcal{B}$ is the set of attainable belief in the POMDP. The initial belief distribution $P' \in \Delta(\mathcal{B})$ over the set of beliefs is given by,

$$P'(b_0) = \sum_{o_0 \in \mathcal{O}} \sum_{s_0 \in \mathcal{S}} \Pr(b_0, s_0, o_0) \tag{B.1}$$

$$= \sum_{o_0 \in \mathcal{O}} \sum_{s_0 \in \mathcal{S}} \Pr(b_0 | s_0, o_0) \Pr(o_0 | s_0) \Pr(s_0) \tag{B.2}$$

$$= \sum_{o_0 \in \mathcal{O}} \sum_{s_0 \in \mathcal{S}} \Pr(b_0 | o_0) \Pr(o_0 | s_0) \Pr(s_0) \tag{B.3}$$

$$= \sum_{o_0 \in \mathcal{O}} \Pr(b_0 | o_0) \sum_{s_0 \in \mathcal{S}} \Pr(o_0 | s_0) \Pr(s_0) \tag{B.4}$$

$$= \sum_{o_0 \in \mathcal{O}} \delta_{f_0(h_0)}(b_0) \sum_{s_0 \in \mathcal{S}} O(o_0 | s_0) P(s_0). \tag{B.5}$$

The transition distribution $T' \colon \mathcal{B} \times \mathcal{A} \to \Delta(\mathcal{B})$ is given by,

$$T'(b_{t+1} | b_t, a_t) = \sum_{o_{t+1} \in \mathcal{O}} \sum_{s_{t+1} \in \mathcal{S}} \sum_{s_t \in \mathcal{S}} \Pr(b_{t+1}, s_{t+1}, o_{t+1}, s_t, b_t, a_t) \tag{B.6}$$

$$= \sum_{o_{t+1} \in \mathcal{O}} \sum_{s_{t+1} \in \mathcal{S}} \sum_{s_t \in \mathcal{S}} \Pr(b_{t+1} | b_t, a_t, s_{t+1}, o_{t+1}, s_t)$$
$$\times \Pr(o_{t+1} | b_t, a_t, s_{t+1}, s_t) \Pr(s_{t+1} | b_t, a_t, s_t) \Pr(s_t | b_t, a_t) \tag{B.7}$$

$$= \sum_{o_{t+1} \in \mathcal{O}} \sum_{s_{t+1} \in \mathcal{S}} \sum_{s_t \in \mathcal{S}} \Pr(b_{t+1} | b_t, a_t, o_{t+1})$$
$$\times \Pr(o_{t+1} | s_{t+1}) \Pr(s_{t+1} | a_t, s_t) \Pr(s_t | b_t) \tag{B.8}$$

$$= \sum_{o_{t+1} \in \mathcal{O}} \Pr(b_{t+1} | b_t, a_t, o_{t+1}) \sum_{s_{t+1} \in \mathcal{S}} \Pr(o_{t+1} | s_{t+1}) \times \sum_{s_t \in \mathcal{S}} \Pr(s_{t+1} | a_t, s_t) \Pr(s_t | b_t) \tag{B.9}$$

$$= \sum_{o_{t+1} \in \mathcal{O}} \delta_{u(b_t, a_t, o_{t+1})}(b_{t+1}) \sum_{s_{t+1} \in \mathcal{S}} O(o_{t+1} | s_{t+1}) \sum_{s_t \in \mathcal{S}} T(s_{t+1} | a_t, s_t) b_t(s_t). \tag{B.10}$$

The reward distribution $R' \colon \mathcal{B} \times \mathcal{A} \to \Delta(\mathbb{R})$ is given by,

$$R'(r_t | b_t, a_t) = \sum_{s_t \in \mathcal{S}} \Pr(r_t, s_t | b_t, a_t) \tag{B.11}$$

$$= \sum_{s_t \in \mathcal{S}} \Pr(r_t | b_t, a_t, s_t) \Pr(s_t | b_t, a_t) \tag{B.12}$$

$$= \sum_{s_t \in \mathcal{S}} \Pr(r_t | a_t, s_t) \Pr(s_t | b_t) \tag{B.13}$$

$$= \sum_{s_t \in \mathcal{S}} R(r_t | a_t, s_t) b_t(s_t). \tag{B.14}$$

Appendix C

Piecewise Linearity and Convexity of the Belief Q-function

In this appendix, we demonstrate the piecewise linearity and convexity of the belief Q-function with a finite horizon [Smallwood and Sondik, 1973], for the Markov decision processes (MDP) described in Appendix B, which we called belief MDPs. Let us assume that, for a given horizon $N \in \mathbb{N}_0$, the Q-function Q_N with finite-time horizon N is piecewise linear and convex in the belief vector,

$$Q_N(b,a) = \max_{\alpha \in A_N^a} \sum_{s \in \mathcal{S}} \alpha(s) b(s), \tag{C.1}$$

where A_N^a is the set of α-vectors for representing the piecewise linear and convex function $Q_N(\cdot, a)$. We thus need $|\mathcal{A}|$ such sets to represent Q_N. Let us now study the Q-function Q_{N+1} with finite-time horizon $N+1$, defined as,

$$Q_{N+1}(b,a) = \mathbb{E}\left[R + \gamma \max_{a' \in \mathcal{A}} Q_N(B', a') \middle| B = b, A = a\right] \tag{C.2}$$

$$= \overline{R}'(b,a) + \gamma \mathbb{E}\left[\max_{a' \in \mathcal{A}} Q_N(B', a') \middle| B = b, A = a\right]. \tag{C.3}$$

The first term is the expected immediate reward in the belief MDP, defined as,

$$\overline{R}'(b,a) = \mathbb{E}[R | B = b, A = a] \tag{C.4}$$

$$= \sum_{s \in \mathcal{S}} b(s) \overline{R}(s,a), \tag{C.5}$$

where $\overline{R}(s,a)$ is the expected immediate reward in the underlying MDP. It can also be noted that $Q_1(b,a) = \overline{R}(b,a)$. This expression is linear in the belief vector, which is a particular case of a piecewise linear and convex function. It can thus be represented with $|\mathcal{A}|$ singletons $A_1^a = \{R(\cdot, a)\}$.

As far as the second term is concerned, let us first notice that the maximum over $|\mathcal{A}|$ piecewise linear and convex functions is also a piecewise linear and convex function, with at most $|\mathcal{A}|$ times more linear pieces. As a result, the value function $V_N(b') = \max_{a' \in \mathcal{A}} Q_N(b', a')$ is given by,

$$V_N(b') = \max_{\alpha \in A_N} \sum_{s \in \mathcal{S}} \alpha(s) b'(s). \tag{C.6}$$

where $A_N = \bigcup_{a \in \mathcal{A}} A_N^a$. Starting from this piecewise linearity and convexity of $V_N(b')$ in b', let us show the piecewise linearity and convexity of $\mathbb{E}[V_N(B')|B = b, A = a]$ in the previous belief b,

$$\begin{aligned}
\mathbb{E}\left[V_N(B')|B = b, A = a\right] &= \sum_{b' \in \mathcal{B}} \Pr(b'|b, a) V_N(b') && \text{(C.7)}\\
&= \sum_{b' \in \mathcal{B}} \sum_{o' \in \mathcal{O}} \Pr(b', o'|b, a) V_N(b') && \text{(C.8)}\\
&= \sum_{b' \in \mathcal{B}} \sum_{o' \in \mathcal{O}} \Pr(o'|b, a) \Pr(b'|b, a, o') V_N(b') && \text{(C.9)}\\
&= \sum_{o' \in \mathcal{O}} \Pr(o'|b, a) \sum_{b' \in \mathcal{B}} \Pr(b'|b, a, o') V_N(b') && \text{(C.10)}\\
&= \sum_{o' \in \mathcal{O}} \Pr(o'|b, a) \sum_{b' \in \mathcal{B}} \delta_{u(b,a,o')}(b') V_N(b') && \text{(C.11)}\\
&= \sum_{o' \in \mathcal{O}} \Pr(o'|b, a) V_N(u(b, a, o')) && \text{(C.12)}
\end{aligned}$$

Using the belief recurrence proof of Appendix A, we have,

$$\begin{aligned}
V_N(u(b, a, o')) &= V_N\left(\frac{O(o'|\cdot) \sum_{s \in \mathcal{S}} T(\cdot|s, a) b(s)}{\Pr(o'|b, a)}\right) && \text{(C.13)}\\
&= \max_{\alpha \in A_N} \sum_{s' \in \mathcal{S}} \alpha(s') \frac{O(o'|s') \sum_{s \in \mathcal{S}} T(s'|s, a) b(s)}{\Pr(o'|b, a)} && \text{(C.14)}\\
&= \frac{1}{\Pr(o'|b, a)} \max_{\alpha \in A_N} \sum_{s' \in \mathcal{S}} \alpha(s') O(o'|s') \sum_{s \in \mathcal{S}} T(s'|s, a) b(s). && \text{(C.15)}
\end{aligned}$$

Substituting equation (C.15) into equation (C.12), we obtain,

$$\begin{aligned}
\mathbb{E}\left[V_N(B')|B = b, A = a\right] &= \sum_{o' \in \mathcal{O}} \max_{\alpha \in A_N} \sum_{s' \in \mathcal{S}} \alpha(s') O(o'|s') \sum_{s \in \mathcal{S}} T(s'|s, a) b(s) && \text{(C.16)}\\
&= \sum_{o' \in \mathcal{O}} \max_{\alpha \in A_N} \sum_{s \in \mathcal{S}} \sum_{s' \in \mathcal{S}} \alpha(s') O(o'|s') T(s'|s, a) b(s) && \text{(C.17)}\\
&= \sum_{o' \in \mathcal{O}} \max_{\alpha \in A_N^{a,o'}} \sum_{s \in \mathcal{S}} \alpha(s) b(s), && \text{(C.18)}
\end{aligned}$$

where $A_N^{a,o'} = \left\{\sum_{s' \in \mathcal{S}} \alpha(s') O(o'|s') T(s'|\cdot, a) \middle| \alpha \in A_N\right\}$, which is a sum of piecewise linear and convex functions. Such a sum is also piecewise linear and convex,

and at any given point, the dominating α-vector could be the sum of any combination of vectors of each of the piecewise linear and convex functions composing the sum. Let us then define the set $A_N^{a,+}$ of sums of every combination of α-vector chosen in $A_N^{a,o'_1}, \ldots, A_N^{a,o'_{|\mathcal{O}|}}$ as,

$$A_N^{a,+} = \left\{ \sum_{o' \in \mathcal{O}} \alpha^{o'}(\cdot) \middle| \alpha^{o'_1} \in A_N^{a,o'_1}, \ldots, \alpha^{o'_{|\mathcal{O}|}} \in A_N^{a,o'_{|\mathcal{O}|}} \right\}. \tag{C.19}$$

From there, we have,

$$\begin{aligned} \mathbb{E}\left[V_N(B')|B=b, A=a\right] &= \sum_{o' \in \mathcal{O}} \max_{\alpha \in A_N^{a,o'}} \sum_{s \in \mathcal{S}} \alpha(s) b(s) && \text{(C.20)} \\ &= \max_{\alpha \in A_N^{a,+}} \sum_{s \in \mathcal{S}} \alpha(s) b(s), && \text{(C.21)} \end{aligned}$$

which is indeed piecewise linear and convex.

Finally, substituting back equation (C.5) and equation (C.21) in equation (C.3),

$$\begin{aligned} Q_{N+1}(b,a) &= \sum_{s \in \mathcal{S}} b(s) \overline{R}(s,a) + \gamma \max_{\alpha \in A_N^{a,+}} \sum_{s \in \mathcal{S}} \alpha(s) b(s) && \text{(C.22)} \\ &= \max_{\alpha \in A_{N+1}^{a}} \sum_{s \in \mathcal{S}} \alpha(s) b(s), && \text{(C.23)} \end{aligned}$$

where $A_{N+1}^{a} = \left\{ \overline{R}(\cdot, a) + \gamma \alpha(\cdot) \middle| \alpha \in A_N^{a,+} \right\}$.