Learning, Prediction and Planning with Approximate Forward Models

Alvaro Ovalle Castañeda

School of Electronic Engineering and
Computer Science

Queen Mary
University of London

Submitted in partial fulfillment of the requirements of the Degree of
Doctor of Philosophy

December 2022

I, Alvaro Ovalle Castañeda, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis. I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

Alvaro Ovalle Castañeda

23 December 2022

Some of the material presented here has also appeared in the following publications:

Ovalle, A., and Lucas, S. M. (2020). Bootstrapped model learning and error correction for planning with uncertainty in model-based RL. In *2020 IEEE Conference on Games (CoG)* (pp. 495-502). IEEE.

Ovalle, A., and Lucas, S. M. (2020). Modulation of viability signals for self-regulatory control. In *2020 European Conference on Machine Learning: International Workshop on Active Inference* (pp. 101-113). Springer, Cham.

Ovalle, A., and Lucas, S. M. (2021). Predictive Control Using Learned State Space Models via Rolling Horizon Evolution. In ICAPS 2021: *Bridging the Gap Between AI Planning and Reinforcement Learning (PRL)*.

Ovalle, A. (2021). An organismic inspired strategy for adaptive control. In ALIFE 2021: The 2021 Conference on Artificial Life. MIT Press.

Other publications not covered in this document:

Bamford, C., and Ovalle, A. (2021). Generalising Discrete Action Spaces with Conditional Action Trees. In *2021 IEEE Conference on Games (CoG)* (pp. 1-8). IEEE.

Abstract

The capacity to build internal representations of the world provides an agent with the opportunity to use them to act in its surroundings more appropriately. These internal representations may capture complex associations and keep track of the state of the agent and the environment. One of the most striking aspects of this phenomenon is that the agent may manipulate these internal representations to consider the distant future, and to formulate plans likely to lead to beneficial outcomes. Our treatment in this thesis considers this particular class of agents referred to as *model-based*. The behaviour of these agents is not only contingent upon the current sensory stream and their memory, but also based on hypothetical future sensory streams that are produced from potential sequences of actions.

Throughout this thesis, there are two main themes that we explore and that are fundamental for advancing our understanding of model-based agents. The first is the agent's uncertainty about its environment and how it influences its decision-making. There are multiple aspects one could investigate about this relation. We analyse two specific scenarios. The first illustrates how it is possible to harness the agent's uncertainty to devise error-correction schemes. In the second, a probability distribution that defines the agent's current model is used to derive intrinsic utility signals to guide behaviour.

The other main theme that permeates this thesis is the question of what are the aspects of the external world that should be stored and represented by an internal model? This question has important consequences for the design of learning objectives. As we will see in this thesis, we start with perhaps the most conceptually intuitive way to frame a learning objective for acquiring a world

model. Namely, the assumption that the agent must be able to predict as accurately as possible its future observations. From this starting point, we progress towards learning objectives that introduce additional prediction targets or constraints to aim for a compressed and more essential representation of an observation. This theme concludes by trying to gain some perspective on whether it is possible, and even desirable, to attempt to have an internal model that tries to map the external observations, as we start to consider information-theoretic notions of relevance. Our results show that these design choices can have a profound effect on performance, even when the planning machinery is identical, and demonstrate the importance of building world models aligned with the agent's behavioural objectives.

Acknowledgments

I would like to begin by expressing my gratitude to my main advisor, Professor Simon Lucas. Throughout my Ph.D. journey and the process of writing this thesis, he provided me with thoughtful feedback, comments, and suggestions. I am immensely thankful for his patience and the freedom he granted me to pursue my research. Additionally, I appreciate his efforts to encourage me to consider the more pragmatic aspects of the work.

I also extend my thanks to the members of my committee, Dr. Jeremy Gow and Dr. Laurissa Tokarchuk, who offered valuable feedback during the Ph.D. I am indebted to my thesis examiners, Dr. Christoph Salge and Dr. Soren Riis, for their rigorous and meticulous evaluations, which greatly enhanced the quality of this work.

I am also grateful to the people within the Game AI group for creating such a friendly and supportive environment. In particular I want to thank Ercument, Martin, and Bamford for the stimulating discussions, the friendship, and the laughter we shared.

I would also like to thank QMUL Research-IT for their technical support and access to the HPC facility, as well as the EECS administrative staff for assistance with bureaucratic matters.

A special thanks to Yumi, for being a source of peace.

Finalmente, quiero agradecer a mi familia por su amor y por su apoyo incondicional. A mis padres, que siempre alentaron mi curiosidad. Este trabajo es para ellos.

Contents

List of Figures

List of Tables

List of Repositories

Chapter 3: Learning, planning, and error-correction with ensembles

https://github.com/aovalle/btf

Chapter 4: Planning in Latent State Spaces with Generative Sequence Models

https://github.com/aovalle/ssm

Chapter 5: Planning with Models of Value Functions

https://github.com/aovalle/qssm

Chapter 6: Planning in Latent State Spaces with Non-reconstructive Forward Models

https://github.com/aovalle/nr-ssm

Chapter 7: Learning and planning from self-regulating signals

https://github.com/aovalle/sr-ainf

Abbreviations

ALE	Arcade Learning Environment
BTF	Bootstrapped Transition Functions
CNN	Convolutional Neural Network
DQN	Deep Q-Network
EFE	Expected Free Energy
ELBO	Evidence Lower Bound
FEP	Free Energy Principle
GRU	Gated Recurrent Unit
iLQR	iterative Linear Quadratic Regulator
KL	Kullback-Leibler
LSTM	Long-Short Term Memory
MAP	Maximum A Posteriori
MCMC	Markov Chain Monte-Carlo
MCTS	Monte-Carlo Tree Search
MDP	Markov Decision Process
MLE	Maximum Likelihood Estimation
MPC	Model Predictive Control
NCE	Noise Contrastive Estimation
NGE	Neural Game Engine
NN	Neural Network
POMDP	Partially Observable Markov Decision Process
RHE	Rolling Horizon Evolution

RL	Reinforcement Learning
RNN	Recurrent Neural Network
RPF	Randomized Prior Functions
RSSM	Recurrent State Space Model
SGLD	Stochastic Gradient Langevin Dynamics
SMiRL	Surprise Minimizing Reinforcement Learning
SSM	State Space Model
VAE	Variational Auto Encoder

Glossary

Active inference: mathematical framework to model sequential decision-making problems and treats perception and action as inferential processes

Autoregressive model: statistical model that predicts a future value based on past values.

Bellman equation: a recursive equation used in several fields, concretely in reinforcement learning it is used to estimate a value function.

Bootstrapping: statistical technique to obtain a sampling distribution based on resampling sampled data.

ELBO: stands for Evidence Lower Bound. Establishes a lower bound on the log likelihood of given data.

Environment: in reinforcement learning, the environment is the medium in which an agent interacts.

Expected free energy: measure for performing inference over an extended action sequence.

Free energy: the information-theoretic notion of free energy refers to the upper bound on surprisal.

Latent state: also known as hidden state. In Bayesian inference it refers to unobserved variables.

Markov Decision Process: mathematical framework to model decision-making problems.

Model: an abstract description of the dynamics of the system used to perform some form of estimation. In this thesis we focus on models that forecast future

values, and we will refer to them as world model, forward model, dynamics model or internal model.

Observation: in this work an observation refers to an agent's sensory input.

Policy: in reinforcement learning, a policy defines how an agent behaves in a given state.

Posterior: updated belief after a new piece of data has been observed.

Prior: initial belief before a new piece of data has been observed.

Reinforcement learning: trial-and-error learning paradigm concerned with finding behavioural policies intended to maximise a reward.

Representation: we are going to refer to representation as the set of features an agent possesses that describe relationships between its sensory inputs.

Rolling horizon evolution: online planning method in which a number of candidate action sequences are evolved and ranked according to heuristic.

State space: set of variables that describe a system.

State space model: probabilistic graphical model that describes the dependencies and relationships between

Surprisal: also known as surprise. In information theory, surprisal corresponds to the inverse of a probability (e.g. $-\log p$).

Value function: an estimate of the expected return from following a policy from a given state.

Variational inference: method to frame Bayesian inference as an optimisation problem.

1 Introduction

There are different strategies to try to solve sequential decision-making problems. Some of these depend on the initial information we have about our task and decompose the larger problem into different underlying subproblems. For example, we can have detailed knowledge about how a system reacts to specific perturbations or how it is modified and changed. We refer more formally to this knowledge as a model, which tells us about the system's dynamics and how it transitions between states. When we have this information the decision-making problem can be solved through planning and search algorithms (Bellman, 1952, 1957; Howard, 1960; Hart et al., 1968; Bertsekas, 1995; Russell and Norvig, 2003; Coulom, 2007). This involves that the chosen algorithm queries the model to produce future trajectories, and because the dynamics are known, the agent can plan and select a course of action that optimises the return.

However, we can also have a different scenario where knowledge about the system's dynamics is not available. In this case, we could, for example, proceed by learning a model from pre-recorded data. This could be limiting when the data is incomplete or biased towards certain regions of the state space. Alternatively, we can opt for a more natural approach and let the agent gather and interact with the environment directly. The latter case is common in online Reinforcement Learning (RL) (Witten, 1977; Watkins, 1989; Rummery and Niranjan, 1994; Sutton and Barto, 1998) in which the agent learns to act through trial-and-error and refines a *policy* to guide its actions by continuously interacting with its environment. Depending on our objectives we can learn both, a model and a

policy, or we could focus exclusively on learning a policy with *model-free* methods. That is, without explicitly considering the transition dynamics.

In this thesis, we take a look at these threads to consider the integration between the planning and the learning problem as well as some of the challenges involved in this integration. We focus on sequential decision-making problems formulated as Markov Decision Processes (MDP). These have been addressed through a variety of methods in dynamic programming and optimal control (Bellman, 1952; Bertsekas, 1995), reinforcement learning (Sutton and Barto, 1998) and more recently active inference (Friston et al., 2012b). In principle *model-based* RL, which has been experiencing a revival sustained by advances in deep learning (for a survey about recent progress please see Moerland et al. (2022) and references therein), combines these two problems. Classical formulations (Sutton, 1991) go as follows. First, the agent accumulates data and creates a dynamics model which then uses to estimate value functions and decide how to act. As the agent continues to solve the task it gathers data to refine the dynamics model and the value functions. Our concern in this thesis, however, is with online planning. We construct agents that sample trajectories from the current step using an approximate model of the dynamics, which then can be evaluated, ranked, and selected. We merge techniques used in the planning community with those used in density estimation and probabilistic modelling in deep learning. A successful example of this merger is MuZero (Schrittwieser et al., 2020) which applied deep learning for model building, value and policy estimation, and Monte-Carlo Tree Search (MCTS) for planning. Despite this, work connecting these two research communities remains relatively rare and has been the focus of recent workshops[1].

Here we follow this direction. We develop new architectures for planning in MDPs in discrete action spaces when the agent does not have knowledge

[1] For example, the planning and reinforcement learning workshop series https://prl-theworkshop.github.io aims to encourage cross-fertilisation between both areas. The work in chapter 4 was presented in the 2021 edition.

of the dynamics. For the planning component, we rely on Rolling Horizon Evolution (RHE) (Perez-Liebana et al., 2013), a look-ahead online trajectory search algorithm. RHE is generic and makes few assumptions, but still requires to commit to a state representation and a heuristic function to evaluate the trajectories. As we will see throughout the thesis, these are two crucial aspects for the integration between the learning and the planning stages.

We are going to continue this introductory chapter by expanding on some reasons why an agent could benefit from acquiring a model, as well as some of the technical and practical challenges involved. Based on this, we will state our research questions and describe the structure of the thesis with our contributions.

1.1 Why to learn a model?

We will approach the sequential decision-making problem from the perspective of the *perception-action loop*, which considers the existence of an agent embedded within an environment where there is a bidirectional and continuous exchange of information through two communication channels: a *sensory* channel necessary to gather data, and an *action* channel that provides the agent some degree of agency to modify the agent-environment dynamics. Essentially the agent uses its actuators to steer a controlled system (e.g. the agent-environment coupling) toward a desirable condition. This could be, for example, to maintain a steady state, the attainment of a goal state, or maximisation (minimisation) of a utility (cost). Then the first question we could start asking is, what is the relationship between the incoming information gathered from the sensory channel and the control exerted by the actuator channel? It is possible to have *open-loop* controllers that do not have an explicit dependence on the current or past states of the environment, that is, they do not sense or acquire information about it. However, the essential character of information as a resource for control, regulation or goal-seeking behaviour has been identified in physics (Maxwell, 1871; Szilard, 1929; Lloyd,

1989; Sagawa and Ueda, 2012; Parrondo et al., 2015), cybernetics (Wiener, 1948; Ashby, 1958; Carver and Scheier, 1981), control theory (Touchette and Lloyd, 2000, 2004; Cao and Feito, 2009) and information-theoretic approaches to the perception-action loop (Klyubin et al., 2005; Polani, 2009; Still, 2009; Little and Sommer, 2011; Tishby and Polani, 2011; Beer and Williams, 2015). In RL, there are different parts involved in information storage. For instance, in the parameters of the policy distribution, a tabular table with Q-values or the weights of a neural network. Model-based RL allows us to be more deliberate about some of the information stored by an agent. As we hinted in the opening paragraph, the model encapsulates knowledge of the task, namely, how it evolves in time and what potential consequences result from an action. What can we gain from treating this knowledge explicitly?

There are several benefits that have been discussed as reasons to design anticipatory agents with models of the dynamics. For instance, it could enable the capacity to forecast events, anticipate consequences, evaluate possibilities, or visualise different perspectives. From a more pragmatic and technical viewpoint, there are also other potential advantages that have been conjectured or are under study:

- **Interpretability:** Similar to other areas in machine learning, the outcomes from the learning systems can often be opaque and inaccessible. Having a model allows us to simulate future trajectories and analyse the actions taken by the agent based on expected consequences estimated by its model (van der Waa et al., 2018).
- **Generalisation:** A major challenge in RL is to avoid overfitting and to generalise to different environments (Cobbe et al., 2019; Farebrother et al., 2020). Applying an agent to a different environment often requires the agent to learn the new task at the expense of the previous one (Kirkpatrick et al., 2017). Research in *transfer learning* focuses on reusing information learned from one task on a different one (Taylor and Stone, 2009; Zhu et al., 2023). It has been speculated that learning a model of the dynamics could

facilitate this transfer by disentangling the interactions between the agent and the objects in the environment (van Seijen et al., 2020).

- **Data efficiency:** A major issue in RL is that algorithms require a vast number of interactions with the environment. Moreover, in some domains, it might be infeasible to collect large amounts of data. With a model, it is possible to train the agent using simulated experience generated from the model without interacting with the environment (Sutton, 1990; Gu et al., 2016).

1.2 The challenges of learning and acting with a model

When we started the work on this thesis, there were some exciting publications where researchers had started to show how to learn dynamics models for large state spaces. For instance, Chiappa et al. (2017) presented a non-interactive learned simulator that predicted multiple steps into the future. Weber et al. (2018) had introduced an architecture combining model-free and model-based paths. The latter is capable of simulating a rollout which is then compressed to provide additional context to the model-free path. Meanwhile, Ha and Schmidhuber (2018) showed an agent capable of training its policy using simulated data. More closely related to our work, was AlphaZero (Silver et al., 2017b). The architecture learned value and policy estimates and combined them with MCTS to plan in board games. Encouraged by these results, our original objective was to use the learned approximate models as foundational tools to explore ideas on intrinsic motivation, rather than becoming a core subject in our investigation. However, when we started implementing our architectures we realised that they were still impractical to pursue our initial ideas. At this point, the architectures had problems dealing with stochastic environments or even with deterministic

moving objects. We also found that initially, we had to use coarse-grained features rather than raw observations, as described in chapter 3. Moreover, we stumbled over issues that had been known in the field. For example, trajectories sampled from a learned model can diverge greatly from the true dynamics, which makes planning over these trajectories essentially meaningless. This is because the model is approximate, subject to updates, and therefore not entirely accurate. This creates a difference between the predictions from the model and the actual outcomes in the environment. These differences are further pronounced as new predictions are generated from the previous ones leading to compound errors (Talvitie, 2014, 2016). In addition, learning a model introduces new sources of instability and error into a training process that is already fickle even without a model (Mnih et al., 2015; Henderson et al., 2019; Dulac-Arnold et al., 2021). Training a model is also more computationally expensive, takes longer to train and the architectures are more complex than the model-free counterparts (Moerland et al., 2022). Furthermore, for online planning, algorithms such as MCTS can be slow as they traverse and visit the states sequentially.

Given this scenario, our research became restructured towards the question of how to integrate learning and planning. This question is broad and naturally involves several other questions. In this thesis, we have decided to take this question and decompose it into a non-exhaustive subset of questions that we try to address.

- **How to plan?:** This first question is related to the mechanics involved in generating the rollouts and how they interact with the approximate model. In our case, we made this question an algorithmic one, and it was motivated by practical reasons. As mentioned briefly in this chapter, we opted for RHE for the planning component. The algorithm is *open-loop*, which allows us to parallelize and evaluate multiple paths. The algorithm also has minimal dependencies from the model, the state representation, and a utility function. This generality then allows us to adapt it to different kinds of models, but it

also leads us to ask:

  - **What representation to use with the planner?:** We are going to refer to *representation* throughout this work as the features an agent possesses that describe relationships between its sensory inputs. It will be a recurrent theme how to define what type of relationships the agent should represent.
  - **How to evaluate the plans?:** We need a way to evaluate and rank according to their perceived worth. This is also learned and predicted by the model. A natural starting point is to evaluate the plans according to their cumulative reward. As we will see we can also use Q-values (chapters 5 and 6) or surprisal (chapter 7).

- **How to deal with inaccurate models?:** not only it is expected that an approximate model cannot fully capture the intricate dynamics of a complex environment. The agent also has to act with a model that is refined continuously because model-learning and decision-making occur intertwined in the training process. This, together with how to choose a representation, forms the central issues in our work. In chapter 3 we show a scheme used to increase the accuracy of the predictions made by the model. However, the fact that the model is inherently inaccurate has several ramifications.
  - **How to handle model uncertainty?:** Rather than trying to make the model's predictions more accurate, we could instead acknowledge the uncertainty in our knowledge and treat it as an element to be modelled as well (chapter 4-7).
  - **How can the state of the model guide behaviour?:** The agent's uncertainty can also be a highly informative signal itself. We show in chapter 7, how an agent can use its model to estimate the current and predicted uncertainty of its future inputs to make its decisions.

1.3 Contributions

The investigations motivated by the questions above resulted in the following contributions:

- In chapter 3 we successfully introduced the first architecture that integrates model-learning with planning via RHE. The novel contributions are the multi-headed model-learning component that learns a predictive distribution of observations based on the statistical concept of *bootstrapping*, three error-correction schemes for prediction, and the integration for online planning.
- In chapter 4 we applied variational methods to learn dynamics models. The models learned using these methods have been used previously to plan in continuous action spaces (Hafner et al., 2019), as pre-trained models for hybrid architectures (Buesing et al., 2018) or in classical approaches to model-based RL (Hafner et al., 2020, 2022). We demonstrated how these models can be learned and used online in discrete action spaces. We introduced the SSM-RHE planner for latent sequences. At a technical level, we improved the planner to execute N rollouts in parallel. We also showed how the architecture can be adjusted to learn local forward models to cope with partial observability on the spatial dimension.
- We extended the previous architecture to learn to predict and simulate value functions. With this new functionality, we presented QSSM-RHE in chapter 5, an online planner that uses either the predicted Q-values or Q-values and rewards to evaluate the rollouts.
- In chapter 6 we derived an expression that is used as a training objective to learn a non-reconstructive model. We also provided a way to approximate this objective via contrastive methods from the representation learning literature (van den Oord et al., 2019). Similar to the previous chapters, we also integrated this novel model-learning component with the RHE planner.

- Within the context of *active inference* (Friston et al., 2016), in chapter 7 we provided some arguments for differentiating between internal variables as a consequence of incoming sensory signals and internal variables produced from interoceptive signals. We argued this separation allows us to introduce a scheme based on homeostatic considerations that give rise to a *preference* distribution that could guide the agent's behaviour in a self-supervised manner. For the planning component, we introduce a RHE planner that evaluates rollout sequences with the *expected free energy*.

1.4 Overview and Thesis Structure

The structure of the thesis is as follows. In chapter 2 we begin with a high-level picture of how to approximate a model, and the basic problem formulation. We also contextualise our work within the broader model-based reinforcement learning literature.

We have hinted that one of the central issues of approximating a model is the role of uncertainty. This might be the result of a model that is still learning but it could also simply be because the model cannot match the complexity of the environment. Naturally, some of the difficulties that concern us are how to accommodate the uncertainty, how to model it or how should it influence decision-making. In chapter 3 we present the first of our investigations where we propose uncertainty-aware RL agents that plan already under the assumption that their models are inaccurate and will lead to erroneous predictions. We propose a scheme to train an ensemble of models that provides: (1) the capacity to aggregate their predictions to increase the robustness of the model, and (2) to obtain a predictive distribution to inform error-correction schemes to improve both, model accuracy and agent performance.

While the architecture in chapter 3 assumes domain knowledge about some of the constraints present in the environment, in chapter 4 we intended to take a

more general approach to the problem of learning a model. We learn to model transition distributions explicitly with the intention of learning the structure of the environment directly. We leveraged the machinery of variational inference to structure the dynamics as a *state space model* (SSM). Amongst other reasons, the move to this probabilistic framework allows us to describe the relationship between a *latent* variable and an observed outcome. For the current context, a latent variable captures essential aspects of the external observations. To integrate it with the planner, although the learning objective is still to reconstruct external observations, the planner receives a latent state or *representation*. Thus, unlike the previous architecture in which the planner operates on the full predicted observations, here the planner acts on a reduced latent space with abstract features that take the place of the observation.

In chapter 5 we extend further the architecture from chapter 4. While in chapter 4 the agents learn the reward dynamics and use them to evaluate and rank potential plans, in chapter 5 we extend the SSM to account for the dynamics of the Q-values. This modification is intended to enhance the criteria used to evaluate the plans. Compared to the rewards, the predicted Q-values provide a richer signal about the desirability of selecting an action over others and provide an estimate of future rewards beyond the planning horizon.

Chapter 6 is motivated by the question of what aspects of the environment should be included in an approximate forward model. In the other chapters we intuitively worked with a learning objective focused on accuracy and reconstruction. However, we could reformulate the objective to be predictive about the future while ignoring as much information as possible from the past. That is, a forward model could attempt to focus on capturing only information that is relevant for the prediction of the future. We will ground this perspective on arguments from information theory, physics, biology, and learning theory.

Chapter 7, chronologically, was developed after chapter 3 but conceptually it

resides at the end of this thesis. We now go beyond the mere integration between model-learning and planning towards a more active involvement of the model to steer the planning and decision-making process. While it is true that the model was already involved, its role was that of a passive provider. It equipped the planner with state and utility predictions. It did not, however, communicate what it is describing in the sense that the actions taken by the agent shape the model to a certain extent. It captures relations between the agent and the environment. As such, the model offers additional information about the epistemic status of these relations. In this chapter, we follow the active inference framework (Friston et al., 2016). From the point of view of utility, active inference grounds it within the statistical regularities of the agent-environment coupling. That is, as a product of the agent's own behaviour as opposed to ascribing it as an abstract property of the environment. We provide a working example of how this utility emerges in a self-supervised manner with a derivation distinguishing exteroceptive from interoceptive signals.

We conclude with some remarks about our contributions, lessons learned, and shortcomings in chapter 8.

2 Approximate Forward Models

2.1 Problem Formulation

The basic scenario that we are going to consider in this thesis is a characterisation of the interaction between an agent and its environment. For the benefit of clarity, at this moment we describe a generic formulation and introduce additional assumptions as required in each chapter. For an overview of Reinforcement Learning (RL) we direct the reader to appendix A.1.

We are going to portray the agent and the environment as discrete-time stochastic processes. Within this context, we assume the existence of an unknown generative process for the environment such that it specifies the rules for how it evolves in time $(..., x_t, x_{t+1}, ..., x_{t+n}, ...)$. The agent interacts with the environment by performing an action a_t at time t sampled from a *policy* π (i.e. $a_t \sim \pi(a|\cdot)$). The environment transitions to its next state x_{t+1} driven by the effect of its inherent dynamics and the agent's action upon it. Through its sensors, the agent gathers some form of feedback. For RL, the standard assumption is the reception of an observation o_{t+1} and a reward r_{t+1} to signal the utility or cost of executing an action in a given state. Depending on the task the observation may correspond to the full state of the environment (i.e. $o = x$) or it might just be a partial view of it.

There are different types of *world models* that an agent can learn. But from the scenario just outlined, a particularly useful model could describe to some extent how the environment-agent interaction changes over time. If this is the case, we

can distinguish three main classes of dynamics models (Moerland et al., 2022):

- Forward model $\hat{o}_{t+1} = f(o_t, a_t)$: Predicts the future observation given an action carried out by the agent and the current observation.
- Inverse model $\hat{a}_t = f(o_{t+1}, o_t)$: Takes two consecutive observations and predicts the action that took the agent from an observation o_t to the next one o_{t+1}.
- Backward model $\hat{o}_{t+1} = f(o_t, a_t)$: It is the counterpart to the forward model. Given an observation, it predicts the previous observation and the action that was taken from it.

In this work, we are going to concentrate on the approximation of forward models. Throughout the document, we will also interchangeably refer to them as world models or internal models. The type of environments that we will consider have a large number of states. Thus, we must employ function approximators such as neural networks, Gaussian Processes (Deisenroth and Rasmussen, 2011) or Gaussian Mixture Models (Agostini and Celaya, 2010) to learn the dynamics. For all our case studies we approximate the forward models through neural networks.

As we have seen from the categorisation, given an action and an observation/state (o_t, a_t), the objective is to learn a mapping to o_{t+1} via a function approximator f_θ,

$$f_\theta(o_t, a_t) = \hat{o}_{t+1} \tag{2.1}$$

We will see throughout the thesis that this mapping can be deterministic or stochastic (i.e. $o_{t+1} \sim p_\theta(o_{t+1}|o_t, a_t)$), depending on how the architecture is constructed. The general idea is to frame model approximation as a supervised (or self-supervised) learning problem. This is facilitated by the data points that are collected by the agent as it interacts with the environment. The high-level objective can then be roughly summarised as the minimisation of the prediction error between the predicted observation $\hat{o}_{t+1} = f_\theta(o_t, a_t)$ and the actual

observation o_{t+1},

$$(f_\theta(o_t, a_t) - \hat{o}_{t+1})^2 \tag{2.2}$$

We are going to consider the approximation of a forward model and its utilisation for planning as simultaneously occurring processes. As the agent learns the forward mappings, it simulates future trajectories contingent on candidate plans which lead to hypothetical interleaved sequences of observations and actions $\tau = (o_0, a_0, o_1, ..., a_{T-1}, o_T)$. Similar to the goal in model-free RL settings[1], the agent must maximise its rewards or equivalently, minimise its costs. The difference to model-free scenarios is that this occurs along the simulated sequences, thus the agent must try to find a trajectory that maximises the expected reward.

2.2 Relevant Work

There is a vast body of work on model-based decision-making. Several techniques could be used to find an optimal trajectory according to the current approximate dynamics model, among the most representative we could include Model Predictive Control (MPC) (Richalet et al., 1978; Garcia and Morari, 1982), Monte-Carlo Tree Search (MCTS) (Coulom, 2007; Browne et al., 2012), iterative Linear Quadratic Regulator (iLQR) (Li and Todorov, 2004), Gaussian Processes (Deisenroth and Rasmussen, 2011), RL (Sutton, 1990, 1991) or through a combination of these methods (Schrittwieser et al., 2020). We will not attempt to cover all the studies done in the area, but instead, we try to provide a contextualisation for the work that we present. Nevertheless, we direct the reader to Moerland et al. (2022) for a recent and comprehensive review of aspects of the application of models, model learning, and planning that extends beyond the scope of the thesis.

[1] Appendix A.1 offers a general overview of RL.

2.2.1 Simulators

We start with what is perhaps the most intuitive manner to learn a forward model. Namely to take a literal interpretation of the forward model ansatz $\hat{o}_{t+1} = f(o_t, a_t)$. As we mentioned previously in the problem formulation, it is possible to interpret all these objectives as supervised learning tasks. Here the percept or sensory data point o_t and the action a_t serve together as an input instance while o_{t+1} is the correct label associated with this input. The case studies and simulations we present in the next chapters are visual-based tasks with high dimensional inputs. The percepts o_t correspond to graphical observations from the frames, either with some form of preprocessing or from raw pixels. Learning the relationships and patterns that exist in these large observation spaces has proved to be challenging. However, with the irruption of the deep convolutional networks and advances in computing power, it became possible to consider this learning problem for raw observations. Previously, some important efforts for modelling high-dimensional visual inputs had consisted, for instance, in taking neighbouring patches from a previous frame to predict a future patch at the next step. Bellemare et al. (2013, 2014) used a *context tree* based approach to generate frame predictions. A major issue with these types of approaches is that there are factors, such as patch size, that are domain dependent and susceptible to the size of the objects or the physics of a particular environment that make these kinds of models ineffective. In Oh et al. (2015), the authors proposed a convolutional neural network architecture that could take the whole frame and thus alleviated some of the constraints from previous efforts. To learn temporal dependencies the architecture takes a sequence of four consecutive frames which, although common practice in RL, is not adequate to capture long-term dependencies. Namely, correlations between non-consecutive frames. The authors addressed this shortcoming in the same paper with the introduction of a second architecture that used a Recurrent Neural Network (RNN). A similar approach was followed in Chiappa et al. (2017) where

the architecture included an RNN to also provide a temporal context for the prediction of a future frame. The work proposed a different training objective. Instead of learning an objective based on next-step prediction, it predicted future steps at three different intervals to try to capture more of the global dynamics of the environment. But let us put into perspective the inclusion of the RNN with the original and simplified forward model relation $(o_t, a_t) \rightarrow o_{t+k}$. The RNN provides additional input from which a future observation is predicted. Thus the mapping is slightly altered to $(o_t, a_t, h_t) \rightarrow o_{t+k}$ where h_t is the RNN's internal state that transports information gathered from previous observations and updated through time. We could increase the generality of this relation to $(c_t, a_t) \rightarrow o_{t+k}$ where c_t simply accounts for the whole context that is available to an agent.

2.2.2 Pretrained Forward Models

Although in essence, the architectures that we just described learned a forward model, these were not harnessed by an agent to plan or learn a policy. Indeed, these architectures did not even model the prediction of the reward signals, an issue that was addressed in Leibfried et al. (2017). The analyses in these earlier works were mostly centred around their prediction capabilities in deterministic environments. In Oh et al. (2015) the experiments also involved verifying whether the trained model could encourage exploration and if it could serve as input for a pre-trained model-free agent. Meanwhile, in Chiappa et al. (2017) the model was used as a simulator of the environment for human play. Another characteristic of these architectures is that the models were learned from static datasets that had been previously generated by pre-trained agents and not simultaneously. However, the breakthroughs and the lessons learned from this body of work motivated a continuation in this direction and led to a subsequent integration between the model and the agent. Similar to the simulators just discussed, these

architectures learned an expert model in advance rather than in conjunction with the agent's actions. A unifying theme in the architectures that we will describe, is that the researchers attempted to investigate how to obtain robust behavioural strategies despite the imperfections in the forward model and the error compounding. This remains an open question, and it is also one of the central points of this thesis. In Racanière et al. (2017), the authors devised an architecture (I2A) that integrated a pre-trained and fixed forward model with a model-free agent to augment its performance. To take an action, the agent queried the model to generate $|\mathcal{A}|$ simulated rollouts, where $|\mathcal{A}|$ corresponds to the cardinality of the actions. Each of the rollouts was initialised with a different action. Then the simulated experience was aggregated to obtain an embedding that was combined to contextualise further the embedding from the current observation to finally be used by the model-free agent. This line of work was continued in Buesing et al. (2018) albeit with the introduction of some major changes. The architecture was structured as a State Space Model (SSM). We will explore in more detail this class of graphical models in chapter 4. For the moment, let us mention that the crucial aspect of this structure is that we can establish a relationship between an observation and a corresponding *latent* state. This is to be understood in a probabilistic context. Latent or *hidden* variables refer to those that are not directly observed. These variables are instead estimated from other observed or measured variables, and they are used to capture properties of a system that are hard to measure In the case of world modelling, this affords us an important interpretation. We can define the latent space to be smaller than the space used by the observed variables. For example, if we use a neural network to encode an image in a much smaller latent space, then its latent representation could be considered a compressed description of the image. Because this latent representation is more compact it could encapsulate essential aspects of the observation if a decoder is trained to reconstruct the latent code back into an observation. In more practical terms, a compact representation can also improve the speed at which the rollouts

occur. Similar to the I2A architecture, the model-free agent aggregated the simulated rollouts in an embedding to provide additional context for value and policy estimation. Another important aspect of this work was the introduction of a stochastic SSM to explicitly model the uncertainty in the transitions and to abandon the assumption of deterministic environments and deterministic forward models. To be more concrete, while other architectures learned a deterministic function $o_{t+1} = f_\theta(c_t, a_t)$, in this case, the architecture learned a density model of the form $p_\theta(z_{t+1}|z_t, a_t)$ from where a latent state can be sampled. There are multiple levels at which the uncertainty of the model can be addressed. In the next chapter, we describe a strategy based on ensembles to account for the uncertainty at the level of the parameters of the model.

2.2.3 Simultaneous Control and Model Learning

The previous architectures considered a strict separation between the acquisition of a forward model and its use for finding good behavioural policies. The two processes occurred detached from each other and it even was assumed the availability of a pre-trained agent that could gather experience for training the model. There are several benefits to gain from following this approach. Learning a model from expert-generated data, especially in smaller state spaces, can promote that the model has adequate coverage and that it learns to capture a wide array of situations. In addition, if we keep these two processes independent it favours their stability. Either of these processes is already a challenging pursuit by itself. Nonetheless, this separation between model learning and control could be artificial and limiting. In complex and large environments it might not be realistic to assume that a model of the world is complete and that it does not need to be updated. Moreover, in most situations, it is not possible to gather expert and diverse trajectories in advance that can be used to train the model. Several recent works have attempted to treat model acquisition and control as interleaved

processes. Ha and Schmidhuber (2018) introduced a highly modular architecture composed of a vision, a memory, and a control component.[2] The vision component used a Variational Auto Encoder (VAE) to capture spatial relationships, for the memory an RNN passed its internal state together with the latent produced by the VAE to model a transition distribution, and finally, a linear single layer network was used to select an action. The authors also showed that the agent could be trained inside the simulated world model. Other architectures such as SimPle (Kaiser et al., 2020), Dreamer (Hafner et al., 2020), Dreamer v2 (Hafner et al., 2022) or Slac (Lee et al., 2020) similarly acquire simultaneously a model and train the agent's policy within the world model which is reminiscent of Dyna-Q (Sutton, 1990, 1991). SimPle implemented an architecture inspired by Oh et al. (2015) with the inclusion of a VAE to estimate the distribution of the next frame to handle stochasticity. To learn the agent's policy it used policy proximal optimisation (Schulman et al., 2017). On the other hand Dreamer, Dreamer v2, and Slac build upon the action conditional SSMs introduced in Racanière et al. (2017). The policies in Dreamer and Dreamer v2 are learned with an actor-critic while a soft actor-critic (Haarnoja et al., 2018) is employed for SLAC.

We have to make a distinction about how an agent could use an approximate forward model. As we have seen in the examples above, in model-based RL it is common to use the model to simulate a trajectory to provide data and estimate a value, or a policy. When the agent acts in the environment, it does not explicitly simulate the future, but compared to the model-free counterparts it follows a policy that has incorporated a more refined notion of what may occur. Another option is to explicitly simulate potential future trajectories and act according to their potential payoff. The PlaNet architecture (Hafner et al., 2019) is an example of this case. The architecture learns an SSM and selects among candidate plans

[2] It is worth noting that the results presented in their paper follow from training the world model (vision and memory components) separately from the controller. However, the authors mentioned that it was possible to train them simultaneously.

via the Cross-Entropy method (Rubinstein, 1999). They tested their approach in continuous motor tasks. Our work in chapter 4 could be considered an exploration of this theme for discrete action spaces.

2.2.4 Implicit Dynamics Models or Implicit Planning

From the existing literature, it is also possible to distinguish a different group of architectures that take a more direct path towards either, the modelling of the interaction between the agent and the environment, or the *look-ahead* decision-making. It is hard to categorise these approaches under a single unifying principle. However a common thread among them, even with those that model aspects of the forward dynamics, is that they are not concerned with modelling the observations of the environment. Instead, these methods opt for the estimation of elements that could exert a direct influence on their action selection. For instance, in Vezhnevets et al. (2016), the agent's policy consists of two modules. An action plan module that for a horizon of size H produces a matrix of probabilities for taking an action at a given time step, and a commitment module that decides whether the agent should continue to follow the current action plan or if it should abandon it and compute a different one. The assumption is that the agent can generate a plan from a single observation at time t and then ignore the rest of the observations until time $t + k$ when its plan-commitment module determines that it should gather and process an observation to update its plan. A different instance of implicit planning occurs in the Value Iteration Networks (Tamar et al., 2016). The learning scheme imposes an inductive bias by feeding a convolutional network with a value and reward map in a recurrent manner to implement an approximation of the value iteration algorithm. Some authors have also speculated that a ConvLSTM could also be implementing a form of implicit planning (Guez et al., 2019). Other architectures such as the Predictron (Silver et al., 2017a), the Value Prediction Networks (Oh et al., 2017), treeQN (Farquhar et al., 2018)

or MuZero (Schrittwieser et al., 2020) learn dynamics models but sidestep the issue of modelling the observations and instead focus on the prediction of RL or planning artifacts that can assist during the decision making. For example, instead of training a network to predict an observation, these architectures are trained to predict the value function, the reward, the policy, or a backup function.

It remains largely underexplored in which scenarios it is convenient to limit the architectures to model exclusively RL and planning artifacts, and in which the observations, even if partially modelled, provide a necessary supervision signal. In chapter 5 we combine the modelling of observations and action artifacts in an agent with explicit planning capabilities. In chapter 6 however, we motivate an information-theoretic treatment of relevance in model-based RL to limit the amount of information an agent should take from the past to predict RL artifacts and their future state.

Architecture	Input Raw Pixels	Temporal Relations	Observation Dynamics	Reward Dynamics	Latent Transitions	Model Used for Control	Simultaneous Learning and Control	Robust to Stochastic Dynamics	Value Estimation	Value Dynamics	Explicit Online Planning
Quad-Tree Factorisation (Bellemare et al., 2013)	–	–	●	◑	–	–	–	●	–	–	–
Skip Context Tree (Bellemare et al., 2014)	–	–	●	–	–	–	–	●	–	–	–
Frame Prediction (Oh et al., 2015)	●	●	●	–	–	–	–	–	–	–	–
Recurrent Env. Simulators (Chiappa et al., 2017)	●	●	●	–	●	–	–	–	–	–	–
Frame-Reward Prediction (Leibfried et al., 2017)	●	–	●	●	–	–	–	–	–	–	–
Imagination Augmented (Racanière et al., 2017)	–	–	●	●	–	●	◑	–	–	–	●
SSM (Buesing et al., 2018)	●	●	●	●	●	●	–	●	–	–	●
World Models (Ha and Schmidhuber, 2018)	●	●	●	–	●	●	◑	●	–	–	–
SimPle (Kaiser et al., 2020)	●	●	●	●	●	●	●	●	●	–	–
PlaNet (Hafner et al., 2019)	●	●	●	●	●	●	●	●	–	–	●
Dreamer (Hafner et al., 2020)	●	●	●	●	●	●	●	●	●	–	–
Dreamer v2 (Hafner et al., 2022)	●	●	●	●	●	●	●	●	●	–	–
SLAC (Lee et al., 2020)	●	●	●	●	●	●	●	●	●	–	–
STRAW (Vezhnevets et al., 2016)	●	●	–	–	–	●	●	–	●	–	●
Value Iteration Networks (Tamar et al., 2016)	–	–	–	●	●	●	●	–	●	–	–
Predictron (Silver et al., 2017a)	●	–	–	●	●	–	–	–	●	●	–
Value Prediction Networks (Oh et al., 2017)	●	–	–	●	●	●	●	●	●	●	●
Deep Repeated ConvLSTM (Guez et al., 2019)	●	●	–	–	–	●	●	–	●	–	–
TreeQN (Farquhar et al., 2018)	●	–	–	●	●	●	●	–	●	–	–
MuZero (Schrittwieser et al., 2020)	●	–	–	●	●	●	●	–	●	●	●
This Thesis											
Chapter 3	–	–	●	●	–	●	–	●	–	–	●
Chapter 7	–	–	●	●	●	●	●	●	–	–	●
Chapter 4	●	●	●	●	●	●	●	●	–	–	●
Chapter 5	●	●	●	●	●	●	●	●	●	●	●
Chapter 6	●	●	–	●	●	●	●	●	●	●	●

Table 2.1: *Architecture comparison. ●: Aspect considered by the architecture. ◑: It is either, mentioned by the authors as possible or inferred from the paper, but not empirically confirmed. – : An aspect not considered by the architecture.*

3 Learning, planning, and error-correction with ensembles

3.1 Introduction

We start with the intuitive notion that to acquire a model of the environment that is useful for decision-making we need a model that is accurate. Where accuracy in this case means how well it can reconstruct the observations. For most tasks, however, learning a faithful model that captures in detail the mechanics of the environment is not attainable. Forward models not only have an epistemic component related to a gap in knowledge that decreases as the agent continues to learn. There is also an aleatoric aspect associated with the stochasticity of the process that generates the dynamics (Kendall and Gal, 2017). Thus for most situations, approximate forward models are inherently incomplete.

One of the most problematic aspects of this situation is that a prediction generated by a model is composed from the predictions it generated before. A model might essentially be useless for planning because even small flaws in the predictions will compound in time (Talvitie, 2014, 2016). Since most research in model-based RL has operated under the premise of learning accurate reconstructive models, recent studies have focused on making the models more reliable. For example, Racanière et al. (2017) designed an architecture with a model-based and a model-free path, compressing a rollout generated by the model based path and using it as additional context for the model-free controller. Buesing et al. (2018) and

Hafner et al. (2019) presented work on learning and executing forward models completely in latent space. Ha and Schmidhuber (2018) showed how adding noise during training can increase the robustness of the models learned by the agent and in Asadi et al. (2019) they proposed a multi-step prediction scheme to increase model accuracy. An alternative approach, sidestepped learning state transitions and instead focused on modelling and simulating other aspects that could help an agent make decisions such as value functions and future policy (Schrittwieser et al., 2020).

In this section, we try to deal with some of the repercussions of planning with imperfect forward models. We depart from the approaches just described. Here we will not be specifically invested in finding improved representation mechanisms intended to learn a more accurate model. Instead, it is assumed that the agent must deal with faulty or incomplete models, and yet, it must plan and act under uncertainty. The method we propose is multi-staged, (1) we apply statistical *bootstrapping* techniques to approximate the predictive distribution of state transitions by learning an ensemble of models through a single multi-headed architecture.

The method we propose applies statistical *bootstrapping* techniques to approximate the predictive distribution of state transitions by learning an ensemble of models through a single multi-headed architecture. We devise and test three different schemes to integrate the data from the distribution. We highlight the importance of having a certain level of introspection into the different predictions made by the ensemble, and how modelling a predictive distribution enables the construction of error-correcting routines to increase the reliability of the predictions. For the error-correcting procedures, we consider that the environment has certain constraints that have to be satisfied by the predictions. If an initial prediction does not satisfy those conditions, alternative predictions contained in the distribution can inform how to adjust the original prediction. We test our approach on a reduced version of Pacman in which the presence or absence of the interacting

elements (e.g. ghosts, power pills) serve as conditions to be fulfilled by the predictions.

3.2 A Probabilistic Perspective on Neural Networks

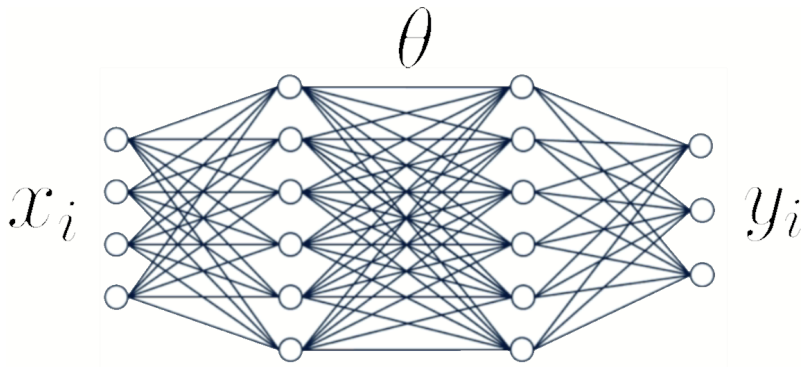


Figure 3.1: *The neural network can be interpreted as a conditional $p(y|x,\theta)$. For MLE and MAP, the networks obtain a point estimate for parameters θ.*

Consider a dataset $D = \{(x^{(i)}, y^{(i)})\}_i^N$ where $x^{(i)} \in X$ is an input (e.g. an image, a pre-processed feature vector, and so on) and $y^{(i)} \in Y$ is a label. A neural network will try to find relationships or mappings between X and Y through parameters (i.e. weights) θ. This can be formally represented as a conditional model parameterised by θ as

$$p(Y|X,\theta) \tag{3.1}$$

typically the objective is to train the network to minimise a loss function to find a set of θ that maximises the likelihood of the data, assuming independent and identically distributed samples,

$$\begin{aligned}\theta^{MLE} &= \text{argmax}_\theta p(D|\theta) \\ &= \text{argmax}_\theta \prod_i p(y_i|x_i, \theta)\end{aligned}$$

this is known as Maximum Likelihood Estimation (MLE). After training, the network has learned the set of assumptions that explains the dataset D given its constraints. MLE tends to overfit and instead, we may decide to include a prior over parameters to regularise them during training

$$\begin{aligned}\theta^{MAP} &= \text{argmax}_\theta p(D|\theta)p(\theta) \\ &= \text{argmax}_\theta \prod_i p(y_i|x_i, \theta)p(\theta)\end{aligned}$$

this type of optimisation is called Maximum A Posteriori (MAP). Also note that if we set the prior to a Gaussian with $\mu = 0$ leads to L2 regularisation (i.e. encouraging small weights), whereas a Laplacian prior would correspond to L1 regularisation (i.e. encouraging sparsity). Using a uniform prior we would recover MLE.

Once training has finished, the network can predict a label y^* given a new input x^* and estimated parameters (θ^{MLE} or θ^{MAP})

$$p(y^*|x^*, D) \approx p(y^*|\theta, x^*) \tag{3.2}$$

MLE and MAP both produce a single point estimate and not a full distribution of θ, thus it is difficult to assess the level of uncertainty in the network predictions. It would be desirable instead of only producing a single set of parameters we could learn a distribution over them to get a more complete view of the distribution of

predictions itself. Prediction with a distribution of parameters would be calculated as

$$p(y^*|x^*, D) = \int p(y^*|\theta, x^*)p(\theta|Y, X)d\theta \tag{3.3}$$

the expression implies that it is necessary to find the posterior $p(\theta|Y, X)$ which involves the following marginalisation

$$p(D) = \int p(D, \theta)d\theta \tag{3.4}$$

as discussed in section A.3 the integral for most cases is intractable and requires us to approximate the posterior using a surrogate distribution $q(\theta|D) \approx p(\theta|D)$. There are a few ways to proceed with the construction of a Bayesian neural network and the approximation of this particular parameter distribution. For example schemes based on variational inference (Blundell et al., 2015; Shridhar et al., 2019), dropout (Gal and Ghahramani, 2016), Stochastic Gradient Langevin Dynamics (SGLD) (Welling and Teh, 2011) and SGLD adaptive preconditioners (Li et al., 2016), or Kronecker factorised Laplace approximation (Ritter et al., 2018).

Researchers have noted that some of these techniques provide poor approximations to the posterior (Osband et al., 2018) or as we have experienced in some unsuccessful instances of our experiments, they might be constrained to static datasets and become unstable in online sequential problems such as those in RL. Nonetheless, the construction of algorithms that can provide better approximations of the parameter distribution remains an active field of research.

3.3 Statistical Bootstrapping

For the work concerning this chapter, we focus our discussion on the potential of *bootstrapping* ensemble methods to approximate the posterior parameter distribution of a neural network. The bootstrap method is a statistical technique intended to assess the accuracy of a statistic or estimator of an unknown population parameter (Efron, 1979). In principle, one could establish the variability of the estimator of interest by repeatedly sampling from the population itself. However, this may not be feasible for situations in which we do not have unlimited access to the population and instead, the only measurements at our disposal come from a unique pool of sample data D. The underlying principle behind the bootstrap method relies on the assumption that this sample data is representative of the true population. Thus we can imagine the process of sampling randomly from the population by treating the sample data as if it was the actual population. If the number of samples is large enough, it can be possible to obtain a distribution that approximates the sampling distribution of the estimator. The process consists of the following steps (figure 3.2):

- From a sample data D of size N resample with replacement to obtain a bootstrap sample $\tilde{D}_k$ of the same size. This process is repeated to get K bootstrap samples.
- Compute a statistic or estimator χ_k^* from each of the bootstrap samples.
- Build a bootstrap distribution with the K estimators.

From the bootstrap distribution then it is possible to perform inference. For example, by calculating confidence intervals or obtaining the standard error to assess the variability across estimates.

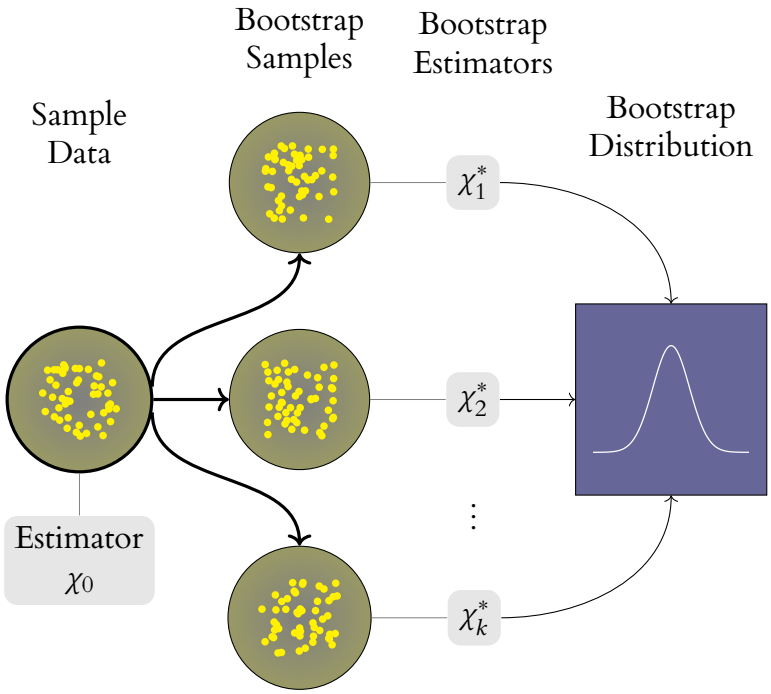


Figure 3.2: *A graphical representation of the bootstrap method. The process starts by resampling with replacement from an original pool of observations (sample data). For each new group of samples an estimator of interest is computed and then used to build the bootstrap distribution.*

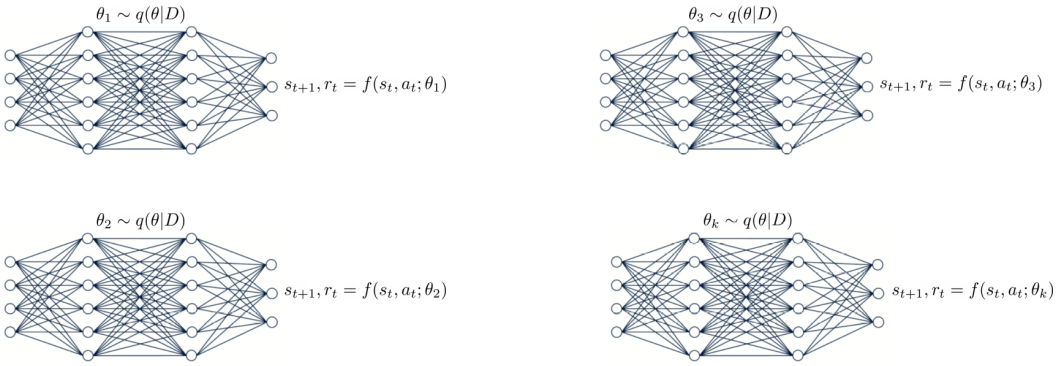


Figure 3.3: *An ensemble model as posterior sampling. Each network with parameters θ_i can be considered as a sample from the approximate posterior $q(\theta|D)$. Gathering the predictions allows us to obtain a predictive distribution of the next states and rewards $p_\theta(s_{t+1}, r_t|s_t, a_t)$.*

3.4 Bootstrapped Transition Functions

We now contextualise the problem of approximating a parameter distribution within RL and more specifically for learning a forward model. Consider the model-based scenario described in appendix section A.1.2 where an agent has to predict future observations and rewards by learning a state transition function given by the conditional $p(s_{t+1}|s_t, a_t)$, and a reward distribution $p(r_t|s_{t+1}, s_t, a_t)$ respectively.

A neural network is trained from data $D = \{s_i, a_i, s_{i+1}, r_i\}_i^N$ corresponding to the transitions observed by the agent where s_i is the state, a_i the action, s_{i+1} the next state and r_i the reward. In analogy to the set-up explained in section 3.2, the input to the network is a concatenation of s_i and a_i and the outputs are the next state s_{i+1} and reward r_i. Formally, we are interested in learning an approximation q of the following parameter posterior

$$q(\theta|D) \approx p(\theta|s_i, a_i, s_{i+1}, r_i) \tag{3.5}$$

to get a predictive distribution

$$p(s_{t+1}, r_t|s_t, a_t, D) \approx \mathbb{E}_{\theta\sim q(\theta|D)}[p_\theta(s_{t+1}, r_t|s_t, a_t)] \tag{3.6}$$

figure 3.3 illustrates intuitively how to think about the samples $\theta \sim q(\theta|D)$. Each of them is a different instance of a neural network parameterised by its own set of θ and producing its outputs. After gathering all these outputs then it becomes possible to construct a distribution and get relevant statistics and measures of uncertainty.

We adapt the bootstrap method described in the previous section to find the approximate posterior $q(\theta|D)$. Training an ensemble of different models separately as exemplified in fig. 3.3 is a time-consuming process, therefore the neural

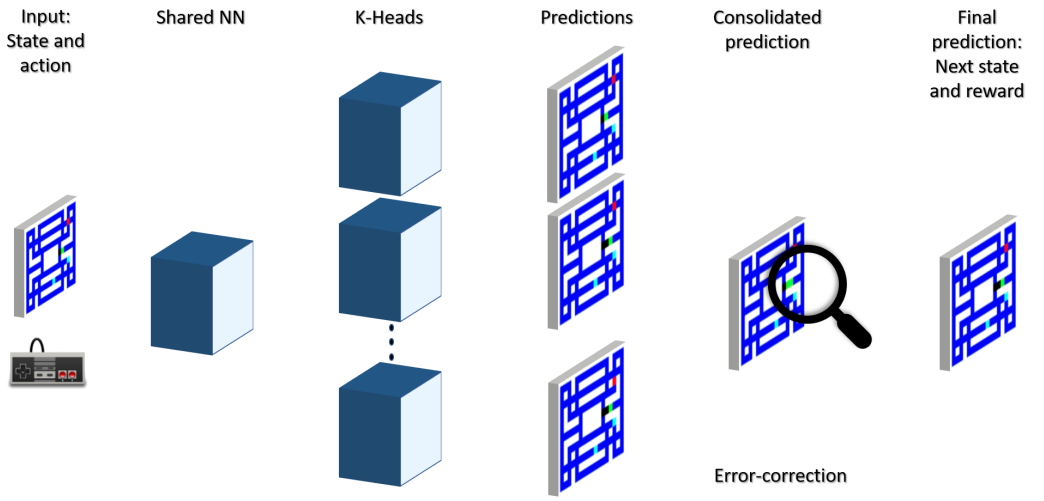


Figure 3.4: *Bootstrapped Transition Functions. The flow starts with a joint input, a concatenation of a frame observation, and an action. This is processed through the multi-headed NN to produce K different predictions to form a single unified prediction (section 3.4.2). The process continues by checking whether the prediction contains errors and if that is the case the prediction is fixed (section 3.4.3).*

network is designed with an architecture bias to train an ensemble of models simultaneously. This is achieved by incorporating multiple output heads in the architecture with each head generating an independent prediction of the next state s_{t+1} and reward r_t (figure 3.4). We refer to our approach as *Bootstrapped Transition Functions (BTF)*. The heads share common parameters in the initial layers of the architecture but crucially each also possesses isolated parameters from the rest of the network. Under this architecture, a predictive distribution of state and reward transitions can be expressed as

$$p(s_{t+1}, r_t | s_t, a_t, D) \approx \frac{1}{K} \sum_k p(s_{t+1}, r_t | s_t, a_t, \theta^k), \quad \theta^k \sim q(\theta | D) \tag{3.7}$$

where θ_k is the combination of the shared parameters of the network and the individual parameters of a head. Therefore we can consider each θ_k as a sample from the posterior parameter distribution $q(\theta|D)$.

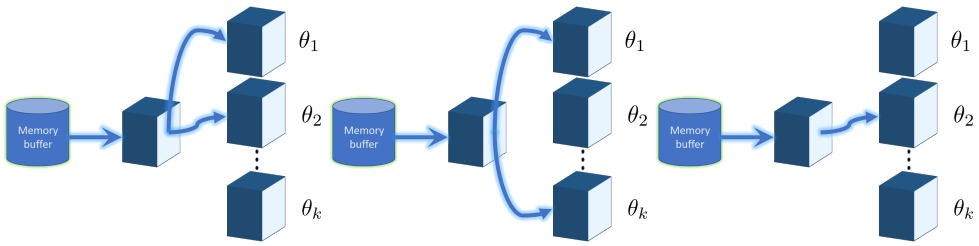


Figure 3.5: *BTF training. A minibatch of transitions is sampled from the experience memory buffer which is used to train one or more of the heads according to a Bernoulli mask. The image illustrates three different instances of the training process.*

3.4.1 Training via Bootstrapping

Throughout their interaction with the environment, the agents store the transitions in an experience buffer. Every time a transition is stored, a Bernoulli binary mask is attached to it to determine on which heads the transition is considered during training and on which it is ignored. Thus to train the multi-headed architecture via bootstrapping techniques, each head is trained with a different group of sampled transitions as indicated by the masks (figure 3.5). Because each head is trained on a different set of transitions, each of them, models an individual approximation of future state and reward. The discrepancy in predictions between the heads is reduced gradually as the training process continues, indicating less uncertainty. Importantly, for new or rare states the heads may predict different outcomes indicating a lack of familiarity and thus more uncertainty in the face of novel situations. After training, the agent can assess the confidence it holds about the predicted next state and reward by sending a new state-action pair through each head and obtaining K different predictions (i.e. the bootstrap distribution). The architecture is trained to minimise a cross-entropy loss between predictions and actual observations or rewards. The whole process is summarised in algorithm 1. Appendix G describes additional details about the architecture.

Algorithm 1: Bootstrapped Transition Functions

Input: Policy π, number of heads K
Initialise: Replay buffer D, parameters θ
for $t \dots T$ **do**
  Sample $a_t \sim \pi(a_t|s_t)$;
  Observe $s_{t+1}, r_t \sim p(s_{t+1}, r_t|s_t, a_t)$;
  if *done* **then**
    $env.reset()$;
  end
  Generate masks $m \sim Ber(K, 0.5)$;
  $D \leftarrow D \cup \{s, a, r, s', m\}$;
  Sample $s_m, a_m, r_m, s'_m, m_m \sim D$;
  $\hat{y} = \{\hat{s}_{t+1}, \hat{r}_t\} = f_\theta(s_m, a_m)$;
  Update model by minimising $\mathscr{L}_\theta(\hat{y}, y)$, backpropagating according to m via SGD;
end

3.4.2 Prediction

As described in the previous section each head produces a unique output which is expressed as $\hat{y}_k = \{\hat{s}^k_{t+1}, \hat{r}^k_t\} = f_{\theta_k}(s_t, a_t)$. But in addition, we also test a small variation of the BTF architecture to include *Randomized Prior Functions (RPF)* (Osband et al., 2018). These are simply neural networks with fixed untrainable but randomly initialised weights. The purpose of including RPF in the output is two-fold: (1) to mitigate one of the shortcomings of the bootstrap method, which by itself does not include a mechanism to reflect uncertainty beyond what it captures from the observed data. With a random network, it is possible to induce artificial diversity in the outputs. (2) The prior acts as a regulariser. The output of the BTF-RPF networks is computed as follows

$$\hat{y}_k = \{\hat{s}^k_{t+1}, \hat{r}^k_t\} = f_{\theta_k}(s_t, a_t) + \beta p_k \tag{3.8}$$

where p_k corresponds to the prior for head k and β is the scaling factor.

To consolidate the different predictions generated by the heads we consider three

different prediction aggregation schemes:

- **Average:** computes the mean of the output values in the last layer of each head and retrieves the class with the largest value:

$$\hat{s}_{t+1}, \hat{r}_t = \text{argmax}_c \left[\frac{1}{K} \sum_k \sigma\left(\mathrm{u}^{(L)}\right)_k \right]_c \tag{3.9}$$

  Where $\mathrm{u}^{(L)}$ is the vector of logits in the last layer L of head k and $c \in \{0, \ldots, C\}$.

- **Majority Voting:** takes the most common prediction by obtaining the mode over the output of each head:

$$\hat{s}_{t+1}, \hat{r}_t = \text{mode}\left(f_{\theta_1}, \ldots, f_{\theta_K}\right) \tag{3.10}$$

- **Sampling:** selects a prediction from the predictive distribution formed by the output of each head:

$$\hat{s}_{t+1}, \hat{r}_t \sim p\left(s_{t+1}, r_t | s_t, a_t; [\theta_k]_{k=1}^K\right) \tag{3.11}$$

3.4.3 Error Correction

There are several ways in which a trajectory simulated by a forward model can deviate from the ground truth. A particular type of error is the lack of cohesiveness between predictions done at the local and global levels. Consider the example of a visual domain, in which a frame can be modelled by a factorised distribution corresponding to individual sections of the frame such as a pixel, a cell, or an object. A network is trained and learns about the representation of various objects in the environment, such as a tree, a car, or a blue cell corresponding to a portion of the sky. If the network predicts the occurrence of a tree in a portion of the frame where only the sky occurs, the prediction of a tree in the frame is

not a surprising event by itself. However, when the prediction is taken globally, the probability of a tree occupying that position is highly unlikely. Using this abnormal and unlikely predicted frame to simulate further into the future may lead to a subsequent string of predictions that degenerate rapidly. Thus in some cases, it is useful to think in terms of the higher-level constraints that need to be satisfied by a prediction.

We have decided to focus on two types of constraints intended to regulate the number of objects of a certain class that should be present in a predicted frame. The first of these constraints is when the frame is missing elements that should be present. The second is when the frame contains multiple instances of an element beyond the number that is expected. We describe how the ensemble can leverage the uncertainty in the estimates via the bootstrap distribution to make predictions that are more robust and with a higher chance of conforming to the expectations of what is realistically possible in the environment. There are three important assumptions. The first is that we know the type of objects that exist in the environment. The second is that before the agent simulates a trajectory, it has an expectation of the number of objects of a certain class that should occur in the environment. The last assumption, is that the agent has memory to access to its previous observations.

To show the potential advantage of having an ensemble to assist error-correction, all corrections made to the frames are based on what is immediately accessible to the agent: (1) its current and past observation and/or (2) its predictions. By having not only a single point estimate but several candidate predictions, we can establish mechanisms that search, compare and integrate observations gathered from the heads into newly revised predictions that are more likely to satisfy the criteria demanded by the constraints. Note that we will refer to an *element* as any generic section in the frame such as a pixel, a cell, or an object.

The generic error-correction flow, as illustrated in figure 3.6, involves five steps:

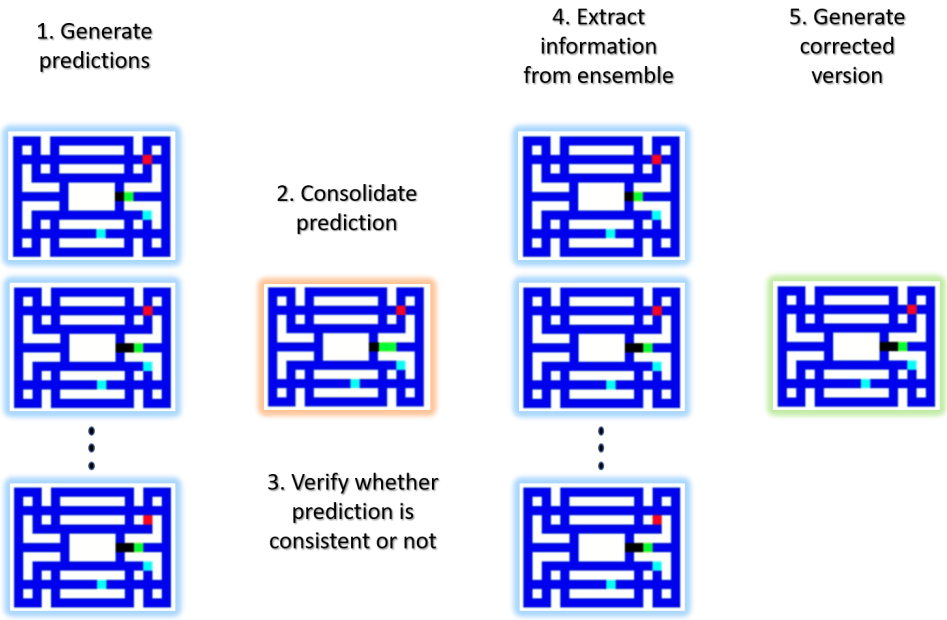


Figure 3.6: *Error-Correction flow. We obtain the candidate predictions from each of the heads to consolidate their information into a single-frame prediction. If this prediction contains errors then the ensemble is used to extract information that can be used to fix it and to generate a final corrected version.*

1. Each head generates a frame prediction.
2. Head predictions are consolidated into a unified prediction as described in the previous section.
3. This prediction is verified to assess if it complies with the constraints required for the prediction.
4. If it fails the conditions then the predictions generated by the ensemble are analysed to extract information that can correct the frame.
5. Finally, a new version of the consolidated predicted frame is generated.

Correcting predictions with missing elements

This type of verification applies to situations where the aggregated prediction does not include the presence of a particular element in any of the cells even though it is expected. To correct the frame, every head is inspected for the element. If a head has one or multiple predictions containing the element, the

positions where the element is located are stored in a shared vector along with the locations coming from other heads. After all, heads have been examined, the frame is corrected by inserting the missing element in a position selected from the shared location vector. The selection is done either by taking the mode of the location vector or by sampling from it, depending on the criteria used by the network to predict the next state and reward. If the location vector is empty because none of the heads predict the existence of the element then the position is taken from the previous frame. Thus only for the first step in a simulated rollout, the previous frame truly corresponds to the last observation gathered from the environment. While for the rest of the steps it is taken from the previously imagined frame. The pseudo-code of this mechanism is outlined in algorithm 2.

Algorithm 2: Error-correction for missing elements

Input: Unified frame prediction f_u, Multi-headed predictions f_{θ_k}, element e, constraint c, last observed or imagined position lp

```
if e in f_u < c then
    Find positions in each f_θk where e is and store in p;
    if p is not empty then
        if sampling then
            p' ~ p;
        else if average or voting then
            p' ← mode(p_0, ..., p_N);
        end
    else
        p' ← lp;
    end
    Insert e in new position p' in unified frame f_u;
    lp ← p';
end
```

Correcting predictions with additonal elements

The other situation we consider is when the prediction includes more elements of a certain type than those anticipated. First, we find the locations where the element is present in the unified predicted frame and store them in a vector **a**. Then we locate the positions of the element in every head of the ensemble

and store them in a shared vector **p**. Similar to the unified frame prediction, a head can also be predicting the existence of the element more times than those specified by the constraints. The two vectors are then compared to identify and separate the locations in the shared vector **p** that occur in **a** as well. From this resulting vector, we select the position of the elements that will be preserved in the unified prediction. The selection is done by either taking the mode or by sampling from the vector.

After it has been determined which elements are preserved, then the next step is to substitute those that have not been selected. This substitution can also be assisted by the predictions contained in the heads. First, by gathering the location of those elements in the unified frame and then, by extracting the predictions done by each head in those locations. At the same time, only the predictions that are different from the element that should be excluded are considered. Once they have been gathered, the selection is again done by taking the mode or by sampling one of the elements. Similar to the *missing element* routine when a decision cannot be made due to an empty vector, the correction defaults to the element occurring in the last frame, whether real or imagined (algorithm 3).

Algorithm 3: Error-correction for additional elements

Input: Unified frame prediction f_u, Multi-headed predictions f_{θ_k}, element e, constraint c, last imagined or observed element le

Initialise: Vector $\mathbf{v}$ that will hold the elements that replace the copies

```
if e in f_u > c then
    Find positions in f_u where e is and store in a;
    Find positions in each f_θk where e is and store in p;
    Check which values in a are in p and store in g;
    if g is not empty then
        if sampling then
            g' ~ g;
        else if average or voting then
            g' ← mode(g_0, ..., g_N);
        end
        Remove g' from g;
        for g_i ∈ g do
            For each f_θk extract the element located at g_i and store in v;
            if v is not empty then
                if sampling then
                    v' ~ v;
                else if average or voting then
                    v' ← mode(v_0, ..., v_N);
                end
            else
                v' ← le
            end
            Insert v' in position g_i in unified frame f_u;
            le ← v';
        end
    end
end
```

3.5 Planning with Ensembles via Rolling Horizon

For harnessing the predictive model acquired by the agent, we introduce an ensemble error-correcting version of Rolling Horizon Evolution (RHE) (Perez-Liebana et al., 2013). RHE is a real-time control and planning algorithm that uses a forward model to search a space of trajectories $\tau = (s_0, a_0, ..., s_T, a_T)$ aiming to find a policy that maximises a utility along the trajectory. A policy π in this case is defined as an action-sequence $a_{0:T}$ of length T.

At each time step, RHE receives the current state s_0 and generates an initial random action sequence which is used to mutate and obtain a population of P action-sequences. The mutator operator selects a step in the sequence with probability p and substitutes the current action with a different one by uniformly sampling the action space. In addition here we apply the *shift-buffer* enhancement introduced in Gaina et al. (2017) that consists of seeding a new population with the fittest action sequence of the previous time step, by shifting it one time step to the left and appending a new random action at the end of the sequence to preserve the length. The fitness of the action sequences is determined by the amount of reward or utility each of them obtains using the forward model until it reaches s_T or an earlier terminal state. The action sequence with the highest total reward along a simulated trajectory is selected. Then, the first action in the sequence is executed in the real environment (algorithm 4). Thus the objective of RHE is to find a policy that satisfies the following:

$$\pi(s_0) = \text{argmax}_{a_{0:T}} \mathbb{E}\left[r(s_0, a_0) + \sum_{t=1}^{T} r(\hat{s}_t, a_t) \right] \tag{3.12}$$

Also note that because this is a stochastic open-loop instance of planning, the

model takes the following form (cf. equation A.8)

$$p(s_0, ..., s_T | a_0, ..., a_T) = p(s_0) \prod_t^T p(s_{t+1} | s_t, a_t) \tag{3.13}$$

although research in RHE typically assumes access to a perfect simulator, the basic mechanics of the algorithm remain unchanged if using imperfect approximate forward models. However, it must be established from where RHE takes the imagined states $\hat{s}$ that it requires to operate. We make two main adaptations, the first is the communication between RHE and the prediction aggregation schemes described in section 3.4.2. In that manner, RHE seamlessly receives the predicted future states and rewards. The second is that the RHE flow is extended to include a condition for error-correction. In that case, the predicted state $\hat{s}$ is fixed and then immediately sent back to RHE.

Algorithm 4: Ensemble Rolling Horizon Evolution with inaccurate models and error-correction

Input: Forward model p_θ, number of actions A, sequence length $\mathcal{T}$, population size $\mathcal{P}$, mutation rate μ

while *true* **do**
  if *not shift buffer* **then**
    $\pi \leftarrow a_t, \ldots, a_{\mathcal{T}} \sim Cat(A)$;
  else
    Shift π to the left and add $a_{\mathcal{T}} \sim Cat(A)$ at the end of the sequence;
  end
  for $i \ldots \mathcal{P}$ **do**
    if $i > 0$ **then**
      $\pi_i \leftarrow$ Mutate π with rate μ
    end
    for $h \ldots \mathcal{T}$ **do**
      Predict $\hat{s}_{h+1}$ and $\hat{r}_h$ using eq. 3.9, 3.10 or 3.11 via p_θ;
      if *error-correction* **then**
        $\hat{s}_{h+1} \leftarrow correct(\hat{s}_{h+1})$;
      end
      Save or update current return $R(\pi_i^{(0:h)}) = r(s_t, a_0) + \sum_{k=1}^{h} r(\hat{s}_k, a_k)$;
    end
  end
  Select fittest action sequence $\pi \leftarrow max_\pi R(\pi)$;
  Perform first action in the sequence $a \leftarrow \pi^{(0)}$;
  Observe $s_{t+1}, r_t \sim p(s_{t+1}, r_t | s_t, a)$;
end

3.6 Experiments

3.6.1 Environment

We assess the effects of the BTF with and without error-correction in Minipacman (Racanière et al., 2017) (figure 3.7). The environment is a reduced version of Pacman and provides a simple minimal discrete control benchmark. The game rules are as follows: an agent navigates through a maze eating food in the corridors. The agent is chased by a semi-stochastic ghost who makes random moves 5% of the time. This active element of the environment introduces an additional challenge for learning a forward model. The agent has to learn the dynamics of an environment that includes elements that it cannot directly control. The agent can also eat *power pills*. These give a 20 time-step period of immunity to the agent and allow it the possibility to eat the ghost. The agent has a repertoire of five actions: up, down, left, right, and no-operation. It also receives a reward depending on specific game events. When the agent eats food it gets a reward of 1. If it does not move or goes to a section of the corridor with no food left, it does not receive a reward. It gets a reward of 3 for the power pills and 6 if it manages to eat the ghost under the effects of the power pills. If the ghost kills the agent the reward is -1 and the game terminates.

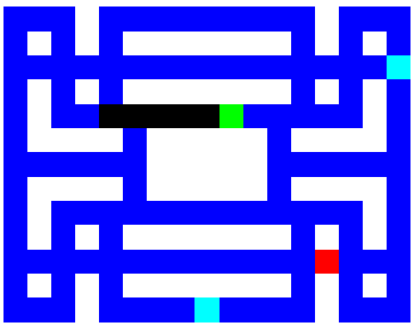


Figure 3.7: *Minipacman. The green cell corresponds to the agent, red to the ghost, turquoise to the power pills, blue to the food still left in the maze, and black to a cell where the food has been eaten.*

Besides the reward, the agent also receives an observation that reflects the current state of the game. The format of the observation is an array of $15 \times 19 \times 3$, where the last number is the number of channels. Thus the observation contains a total of 15×19 cells. Each of them is associated with a different object: an agent, food, power pills, a ghost, eaten cells, and inaccessible cells. The fact that we can treat a cell as an object allows us to evaluate our error-correcting strategy as proof of concept since unsupervised object detection for control tasks is still a nascent area of research (Kulkarni et al., 2019; Watters et al., 2019; Anand et al., 2020).

3.6.2 Error-Correction in Minipacman

To test the performance of the proposed error-correcting algorithms, we focused on the most fundamental elements of the game: the agent and the ghost. Predicted frames are checked against the constraints established for the game. The constraints are simply the number of agents or ghosts counted from the last real observation (or the last imagined observation). If an element is absent or there is more than expected, the routines in algorithm 2 and algorithm 3 are invoked respectively. However, there is a small element-specific adjustment done for algorithm 2. If the predicted frame initially contains more than a single Pacman, then from the pool of copies to be discarded, it is verified if one of them corresponds to the position where it was observed in the last frame. If the condition is met then the cell is turned black, symbolising that the food there has been eaten. Otherwise, they follow the original outline and take their values from the last observed or imagined frame.

3.6.3 Model Learning

The first empirical evaluation considers the capacity to predict future state and reward. We compared a non-probabilistic single-headed agent against six variations of BTF agents. Three of the agents operated with a forward model

trained through standard BTF while the other three also included a random prior (BTF+RPF). For both groups, we tested the three mechanisms to consolidate predictions defined by eq. 3.9, 3.10 and 3.11. All CNNs were trained for 50,000 steps using a random policy and then tested on their next step predictions over 100 episodes. The accuracy was measured by verifying the number of cells (for objects) or reward class, that matched with the ground truth.

If we observe the upper section of table 3.1, the average accuracy in the entire frame makes can be deceiving as all models seem to be close to each other. This is because most of what the models have to predict has to do with the general structure of the environment, namely the inaccessible cells and the corridors. Nonetheless, the differences between them are easier to appreciate by decomposing the accuracy by element category. Uncertainty-aware agents are consistently able to predict the motor consequences of their actions to a larger extent than when using a single-headed model. Although agents using ensemble models predict Pacman position with great accuracy, all agents are less adept at predicting the semi-stochastic behaviour of the ghost. Activating the error-correcting routines has a positive effect on both categories. They allow the agents to predict correctly the ghost movement up to 50% of the time.

Error-correcting mechanisms however provided something considerably more critical. From the observations we had gathered on model prediction it was not rare to find frames where Pacman was missing or cloned, hence justifying the error-correction mechanisms proposed in this chapter. This was also a common occurrence for the ghost. The bottom section of table 3.1 (i.e. Constraint Fulfilment), refers to the proportion of frames that complied with the element constraints imposed by the environment. Namely, the predicted frames that did not miss or included extra elements compared to the real observation. There we can observe that error-correcting mechanisms accurately included Pacman and increased significantly the likelihood of removing or including the exact number of ghosts that appear in the next time step.

	Single Head	*Average*		*Majority Voting*		*Sampling*	
		Boot	***RPF***	***Boot***	***RPF***	***Boot***	***RPF***
Accuracy							
Fruit	0.99583	0.99627	0.99617	0.99627	**0.99638**	0.99617	0.99617
Eaten cell	**0.99598**	0.99545	0.99261	0.99564	0.9926	0.99506	0.99174
Pacman	0.90284	0.99899	0.99899	0.99919	**0.99939**	0.99878	0.99777
Pacman (EC)	-	0.99939	0.99979	0.99939	**0.99979**	0.99939	0.99959
Ghost	0.22980	0.36043	**0.47513**	0.35595	0.46303	0.34722	0.40053
Ghost (EC)	-	0.49955	0.50134	0.49955	**0.50179**	0.49865	**0.50179**
Reward	0.9557	0.97356	0.97376	**0.97519**	0.97437	0.97112	0.97397
Frame	0.9962	0.99661	0.9966	**0.99663**	0.99661	0.9966	0.99658
Constraint Fulfilment							
Fruit	0.86594	0.96433	0.98296	0.95127	**0.98566**	0.9548	0.97173
Pacman	0.90499	0.95564	**0.99768**	0.99654	0.99584	0.97429	0.99558
Pacman (EC)	-	1.0	1.0	1.0	1.0	1.0	1.0
Ghost	0.31941	0.63119	0.85846	0.63407	**0.86749**	0.51388	0.7348
Ghost (EC)	-	0.92741	0.9197	0.93	**0.95096**	0.94887	0.92379

Table 3.1: *Model learning. Top: Average next step accuracy by category and type of forward model. Bottom: The proportion of frames that satisfied exactly the constraints as observed in the environment. The results report a single-headed agent and the multi-headed BTF agents using different aggregation schemes and network output. We can observe how the architectures that include the error-correction schemes (EC) significantly increase the capacity to fulfil the predictions according to what is expected by the actual environment's dynamics.*

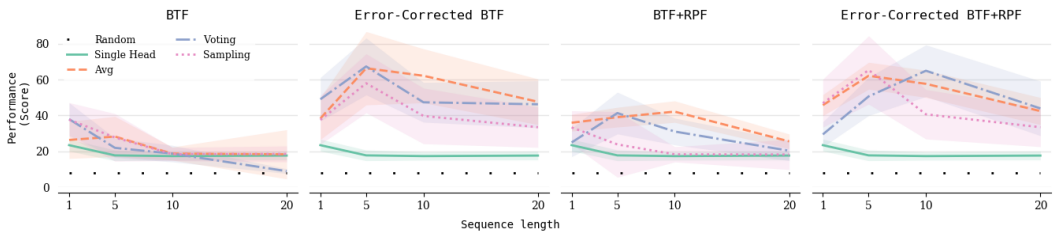


Figure 3.8: *Comparison of uncertainty-aware BTF agents against a non-probabilistic agent with a single-headed model. The plots show the performance in Minipacman with different planning horizons.*

3.6.4 Planning and game performance

We measured the performance of the agent in the game to analyse the impact that the probabilistic models have when planning with simulated temporal sequences. After the models were trained they were tested in combination with RHE for several sequences as specified in table B.1 and evaluated for 10 episodes. We observed that in general all bootstrapped models, given a choice of sequence length, offered better performance than a non-probabilistic single-headed model (figure 3.8). At their peak, bootstrapped models without error-correction achieved a $2x$ improvement and above $3x$ when enhanced with error-correction. All models degraded in performance as horizon depth increases, however error-correcting models tended to benefit more from longer sequences and managed to retain proficiency even with 20-step sequences. For reference, RHE using a perfect simulator achieves a mean score of 134.54.

We also noticed that BTF+RPF exhibits a more stable performance than BTF alone. Combining these observations with those from the results in model learning, it is possible to speculate that the prior is indeed acting as a regularisation mechanism. This could imply that during training the ensemble is more diverse in its predictions and therefore less susceptible to premature overconfident estimations.

3.7 Discussion

The objective in this chapter was to investigate the potential benefits in performance that learning a predictive distribution could grant us when we aggregate the samples. Although in the literature there has been a large emphasis on producing more accurate models, our results support conclusions reached in previous work regarding the shortcomings of pursuing model accuracy for its own sake (Talvitie, 2014). Producing highly accurate immediate predictions may still lead to catastrophic failure if the accuracy measures we use are global and fail to capture how the essential components of the environment are modelled. Thus there is a necessity to establish and study protocols to assess the quality of a model that extends beyond global accuracy. Here we focused on evaluating the structural consistency in the predictions given by the elements and objects that are relevant to the dynamics of the environment. Compared to single prediction agents, the results demonstrated that training a forward model capable of producing multiple estimates *as if* they were samples from a distribution, leads to improved performance. Moreover, we showed that it is possible to use these samples and use them to produce an unified prediction enhanced by error-correction schemes increasing stability in longer horizons. On a conceptual level, this may suggest that frame coherence could be complementary to (granular) accuracy for correctly planning in uncertain environments.

Although BTF without error-correction can be generically applicable to multiple domains, BTF with error-correction is provided as a proof of concept, as it is still limited in scope. There are several directions to reduce these limitations. For instance, we have assumed prior knowledge of the constraints. It could be possible to learn these constraints directly from data by training another neural network to predict the presence or absence of elements according to the game transitions and then to pass them to the error-correcting routines. Alternatively, as we are going to see in the next chapter, a different route to generalise is to

model explicitly the transition distribution.

4 Planning in Latent State Spaces with Generative Sequence Models

4.1 Introduction

In the previous chapter we considered an architecture in which the observations undergo a preprocessing step before they are passed on to the planner. In Minipacman, the observation corresponds to colour-coded categories such as a ghost, the agent, or empty space. The assumption is that the environment is represented by cells corresponding to specific objects. This makes adapting the error-correction schemes to other environment not straight-forward.

We can try to overcome these constraints and increase the generality of the architecture by considering learning from raw pixels. Until recently, this proved to be far from trivial, research on learning predictive models from raw inputs has followed a progression starting from next frame prediction (Oh et al., 2015; Chiappa et al., 2017) and joint frame-reward prediction (Leibfried et al., 2017) where the results showed that it was possible to maintain a low prediction error over long time-spans using modern deep learning architectures. Although these results were obtained in non-controllable supervised settings further research led towards combining the usage of a learned forward model to acquire a policy (Ha and Schmidhuber, 2018; Buesing et al., 2018; Holland et al., 2019). In Holland et al. (2019), the authors built upon the convolutional network from Oh et al. (2015) integrating the imperfect learned model with a Dyna-style agent (Sutton,

1990). The agent updates the value function using not only real experience but also simulated experience generated by the model. While this agent learned temporal patterns by receiving a sequence of stacked frames as input, Ha and Schmidhuber (2018) took a different approach. Their architecture consists of a Variational Auto Encoder (VAE) in charge of identifying spatial patterns and a Recurrent Neural Network (RNN) for extracting codes from temporal sequences. The controller is relatively small and learns a policy on top of the other two components. The authors showed that the controller was even capable of learning a successful policy using only simulated experience. Notably, this architecture and the slightly earlier approach presented in Buesing et al. (2018) modelled *latent* state transition distributions. The concept of a latent variable is crucial for this chapter and will also be used throughout the rest of the thesis. In a probabilistic context, a latent or *hidden* refers to those that are inferred from observed variables that can be directly measured. The application of sequential latent variable models brought two major consequences: (1) the agents were better equipped to use error-prone imperfect models as the uncertainty was explicitly modelled to a certain extent, and crucially (2) the latent variables were used to reduce the dimensionality of the observations. The latent space is specified to be smaller than the pixel-space that defines the observations. Thus several observed variables are aggregated to single latent variables. This has the effect that, for instance, if the training objective is to reconstruct an observation, the latent space may store essential features necessary to reconstruct it. Accordingly, it also discards a lot of the data from the observation as the latent space is designed to be a compressed version of it. This also has practical advantages, the sequential latent models allowed model-based architectures to predict next states in the smaller latent space, increasing the speed at which the predictions can be generated. For our purposes, this means we can generate and evaluate a larger number of plans with the online planner.

Here we try to address a major shortcoming from the architecture presented in the

previous chapter. We leverage the advancements in sequential latent modelling to construct an architecture that processes the environment from high-dimensional images, thus avoiding preprocessing or feature handcrafting. Similarly to the agents proposed in the previous chapter we approach this model construction from a probabilistic perspective as a natural way to reflect the uncertainty of the future. However, unlike in Chapter 3 the uncertainty that is modelled is not about the model parameters to produce a transition but about the state transition. The sequential latent model will be represented as a State Space Model (SSM) (Kalman, 1960; Kalman and Bucy, 1961), a class of probabilistic graphical models that describes the dynamics and relations of the different time series. As we will see, with an SSM we could describe historical relationships, where a variable carries information into the present from other variables from the past. This could afford us the possibility to provide additional context to the controller beyond just the current state.

As reviewed in chapter 2, the SSMs have been used within the context of Reinforcement Learning (RL) in Buesing et al. (2018) to learn to predict sequences and aggregate them as part of the context given to a policy, Hafner et al. (2019) used an SSM to learn models of tasks in continuous action-spaces, Hafner et al. (2020, 2022) showed a model-based agent which learned a forward model and actor-critic, while Lee et al. (2020) learned the same components with a hierarchical SSM. Our two main contribution in this chapter are (1) to extend the usage of approximate models learned with a SSM to online planning in discrete action-spaces. This involves providing an interface to the Rolling Horizon Evolution (RHE) planner to communicate with the SSM. Then from within the planner we can query the SSM to traverse and produce simulated latent sequences on one hand, and to predict the return of those sequences on the other. From the previously cited works, Hafner et al. (2019) architecture is the only one that performed online planning. In this regard, the architecture we present in this chapter could be considered as its counterpart in discrete action-spaces. (2) We

also show how our architecture is capable of learning local forward models, and plan with them, by limiting the agent's inputs to its immediate neighbourhood.

Because the SSM takes a prominent role in this and the next two chapters, we are going to proceed in the next section describing how do we go from an autoregressive class of models, such as the architecture described in the previous chapter, to a SSM capable of modelling stochastic dynamics and transporting information from the past (i.e. Recurrent State Space Model (RSSM)).

4.2 Model Taxonomy: from raw to latent spaces

We have previously considered learning a world model from single next-step transition tuples (s, a, s'), however, focusing exclusively on a single transition neglects the information content that is communicated throughout a sequence. Moreover, it also becomes inefficient if the task is partially observable due to incomplete or noisy observations.

4.2.1 Autoregressive models

Stated simply, an autoregressive model predicts an outcome based on past values. Let us consider the following problem formulation. Assume that we want to model a distribution for the next observation, but rather than simply modelling $p(o_{t+1}|o_t, a_t)$ we take a (sub)history of past observations $o_{\leq t}$ and actions $a_{\leq t}$ to compute $p(o_{t+1}|o_{\leq t}, a_{\leq t})$. We could imagine a function f that maps this history to an output such that $p(o_{t+1}|f(o_{\leq t}, a_{\leq t}))$. The graphical model that would correspond to this configuration is depicted in fig. 4.1, among the major issues with this type of model, is that often they do not reuse previous computations when evaluating an $f(o_{\leq t}, a_{\leq t})$ and that the observations have to be reconstructed to roll out a sequence.

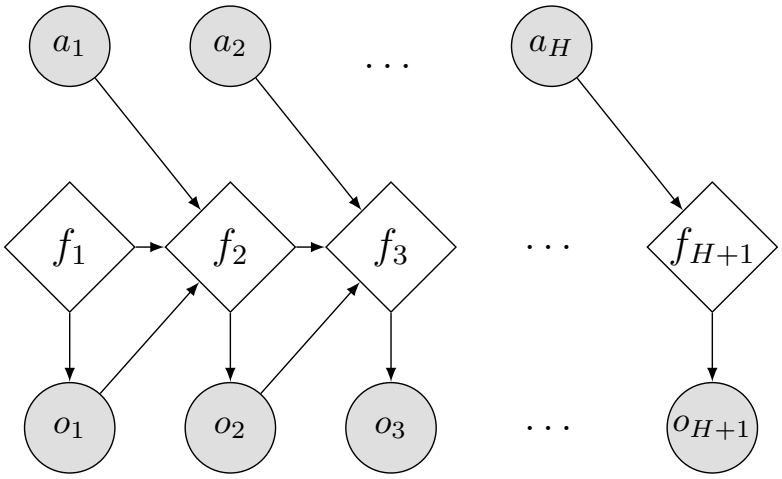


Figure 4.1: *A graphical representation of an autoregressive model. The diamond-shaped nodes and the circles indicate deterministic functions and random variables respectively.*

4.2.2 Deterministic State Space Model

Given some of the constraints of the vanilla autoregressive models, we could consider an alternative approach to model the sequential time series. The SSM (Kalman, 1960; Kalman and Bucy, 1961) invokes the notion of a *latent* or *hidden state* (i.e. variable) that is assumed to capture the essential aspects from other observed variables. From the point of view of world modelling this can be framed as an agent that collects an observation from which it could infer a latent state. This latent state will represent an abstraction compressing some of the elements from the environmental observations. We can go a step further and make the latent state compress not only the current observation but also the history, working as a form of memory. In practice, we could parametrise a neural network, such as an RNN, to recursively map (i.e. encode) in a deterministic way the previous latent state and last observation to an updated latent state as $s_t = f(s_{t-1}, a_{t-1})$ (Cho et al., 2014b). In fig 4.2. we illustrate this process. We can observe that the latent state s_t acts as a sufficient statistic from which we can predict o_t. In this manner, the state space model becomes less computationally demanding because the observations do not have to be reconstructed as the state

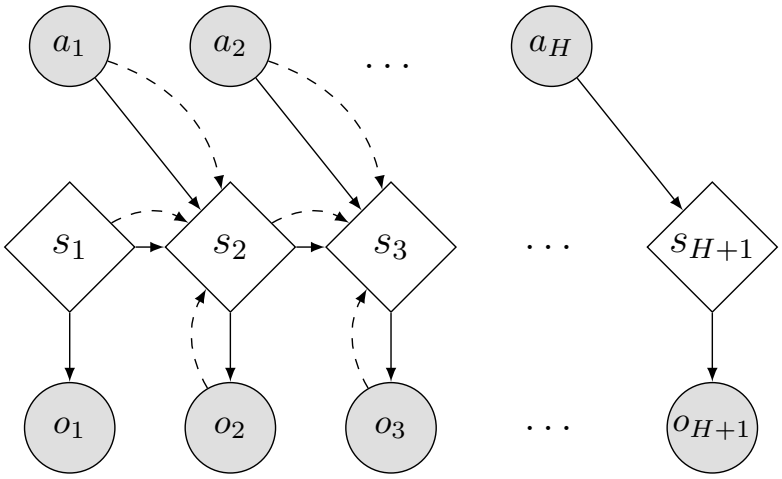


Figure 4.2: *Deterministic state space model. The deterministic diamond shapes nodes represent hidden states such as those modelled by a recurrent neural network. The solid and dashed lines indicate the generative and the inference processes respectively.*

transitions necessary to simulate a sequence occur in the more compact latent space.

4.2.3 Stochastic State Space Model

A different variant of the SSMs considers our interest in modelling the transition distribution probabilistically to account for the uncertainty over a latent state s. Effectively that means that instead of having $s_{t+1} = f(s_t, a_t)$ we now have a distribution $p(s_{t+1}|s_t, a_t)$ from where we can sample the next state s_{t+1}. Accordingly, as observed in fig. 4.3 if we were interested in predicting o_t we would do so through a hidden state s_t, that is $o_t \sim p(o_t|s_t)$.

4.2.4 Recurrent State Space Model

Some empirical analyses have speculated about the tradeoffs between the deterministic and the stochastic SSM. Racanière et al. (2017) argued that while a deterministic SSM cannot model joint uncertainties in time or space, the stochastic SSM was limited to capturing it over pixels but not over time steps. Hafner

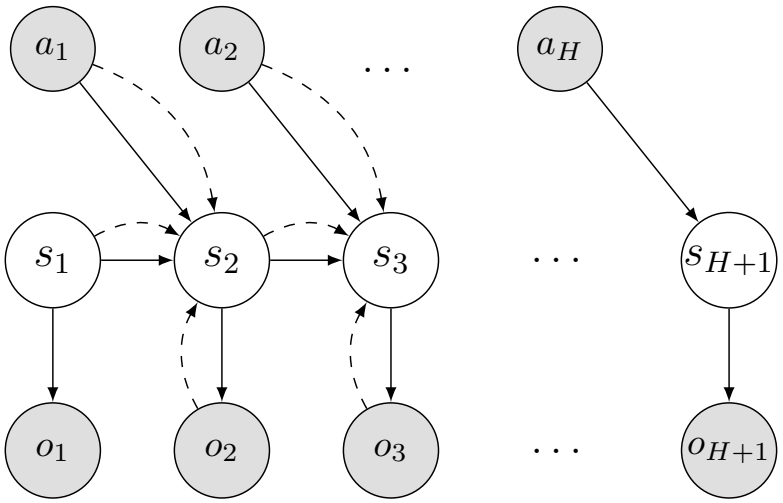


Figure 4.3: *Stochastic state space model. Unlike in deterministic SSMs, the hidden state s is represented by a random variable.*

et al. (2019) provided additional support for this argument. They speculated that due to the aleatoric nature of the stochastic SSM it was difficult to transport information over several time steps. For the deterministic SSM they observed that it could hold information over multiple time steps but it could not capture multiple timelines and as a consequence, it could not learn. However some of these conclusions have since been disproven, Lee et al. (2020) successfully implemented a hierarchical stochastic SSM. The authors argued that by sampling from the posterior as opposed to the prior, it was possible to control the sensitivity during the propagation of the latent states. It could also be speculated that due to the hierarchical structure of their model, the higher expressiveness increased the capacity to capture more of the dynamics.

Another option to deal with the trade-offs between the stochastic and the deterministic SSM is to integrate them into a single architecture (Hafner et al., 2019). This is illustrated in fig. 4.4 where the latent state is composed by a deterministic h and a stochastic s component. This combination allows us to store information from the past via the RNN while considering the different transitions that could occur with the stochastic component, to learn to predict multiple alternative

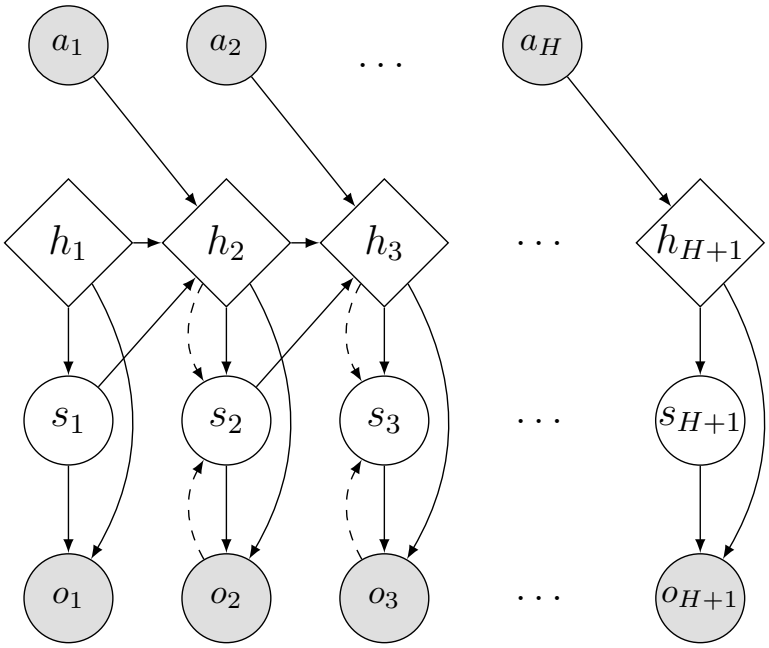


Figure 4.4: *Recurrent stochastic state space model. This SSM incorporates deterministic (h) and stochastic (s) routes within the same architecture.*

timelines. This variation is known as the Recurrent State Space Model (RSSM) and forms the basis of the model-learning component in this chapter.

4.3 Learning a State Space Model

4.3.1 Dynamics and predictions

We are now in the position to describe a general strategy on how to learn a forward model structured as an SSM. We can divide the problem into two parts, one corresponding to learning how to represent the agent's sensory inputs and memory, that is the latent state, and a second sub-problem related to learning the dynamics of the model to generate predictions. For the latter we require two basic ingredients:

- Transition model $p(s_{t+1}|s_t, a_t)$: where s corresponds to a latent state. It helps us to traverse forward in time to reach the next latent state.

- Target model $p(y_t|s_t)$: where given the information contained in a latent state s we can predict something of interest. This does not have to be constrained to a target at the current time step y_t but could also be for targets further in time (e.g. $p(y_{t+k}|s_t)$). For the agents in this chapter we are going to consider two target models:
  - Observation model $p(o_t|s_t)$: as we will see below in the next section, its role is to serve as the main supervision signal. During training time, the architecture produces a reconstructed observation to compare against the actual observation. Because the reconstructed observation is conditioned on what the agent has learned to represent in s it is going to be encouraged to capture essential elements from the original observations.
  - Reward model $p(r_t|s_t)$: the reward model provides an additional learning signal to help to represent relations between the observations, actions, and rewards. It also fulfils a crucial role during planning where the agent simulates a sequence of latent states. From the latent states, the agent is then able to predict the potential rewards from executing hypothetical action plans.

4.3.2 Representation

To learn an adequate representation s that captures useful Spatio-temporal patterns it is necessary to consider the input sources that are available for information extraction. In fig. 4.3 we illustrated a general design for the dynamics of the model where we observed that at time t a latent state s_t depends on the previous s_{t-1}, that transports information from the past, as well as the current observation o_t and the action a_t that lead to that observation. We could then learn a distribution $p(s_t|s_{t-1}, a_{t-1}, o_t)$, also known as the *filtering distribution*, that specifies these dependencies to infer s_t.

4.3.3 Objective

Let us remember that our original interest is modelling $p(o_{t+1}|o_{\leq t}, a_{\leq t})$. Generalising this distribution for future observations along a trajectory and considering an SSM it can be factorised as:

$$p(o_{1:T}|a_{1:T}) = \int \prod p(s_t|s_{t-1}, a_{t-1}) p(o_t|s_t) ds_{1:T} \tag{4.1}$$

Where $1:T$ indicates a sequence from 1 to T. There are two aspects to notice here. We can train the model by performing maximum likelihood estimation, however, the evaluation of the distribution is intractable and therefore we have to approximate it. Second, a possible way to interpret the two parts, the representation, and the predictive part, is as an encoder-decoder architecture and more concretely a sequential VAE. An option is then to maximise the Evidence Lower Bound (ELBO) with variational methods to approximate the target distribution:

$$\ln p(o_{1:T}|a_{1:T}) \geq ELBO = \sum_{t=1}^{T} \mathbb{E}_q[\ln p(o_t|s_t)] - \mathbb{E}_q[D_{KL}[q(s_t|o_{\leq t}, a_{<t})||p(s_t|s_{t-1}, a_{t-1})]] \tag{4.2}$$

For a review on variational inference we refer the reader to appendix A.3

4.3.4 Training details

Thus far we have described a very general overview of the components and the learning objective to acquire a forward model structured as an SSM. There are multiple valid practical specifications and design choices to decide on how to train it. The first point to emphasise here is that the type of SSM we are going to be training is the RSSM. Therefore as we observed in fig. 4.4 we have to split the latent state into a stochastic and deterministic component, that is $\{s, h\}$. We

have discussed in the previous sections that in non-hierarchical architectures the deterministic component tends to be more involved in the storage and transport of information across time steps. An option for implementing this deterministic component is with an RNN (Cho et al., 2014b). We use a Gated Recurrent Unit (GRU) (Cho et al., 2014a) but in principle, any module capable of handling long-range context dependencies could be used. For instance, an architecture similar to ours has been recently proposed where the deterministic component is implemented with a transformer (Chen et al., 2022). For our architecture then we end up with the following major components:

- Encoder $q_\phi(s_t|h_t, o_t, a_t)$
- Deterministic transition $h_t = \text{GRU}_\theta(h_{t-1}, s_{t-1}, a_{t-1})$
- Stochastic transition $p_\theta(s_t|h_t)$
- Observation model $p_\lambda(o_t|h_t, s_t)$
- Reward model $p_\xi(r_{t+1}|h_t, s_t, a_t)$

This is similar to the RSSM proposed in (Hafner et al., 2019). Note, however, that for the reward model we do not predict the reward obtained when a given latent state $z_t = \{h_t, s_t\}$ is reached, that is $p(r_t|h_t, s_t)$. Instead we found an increase in performance and stability if we predicted the reward that could be obtained from this latent state when executing action a_t, namely $p(r_{t+1}|h_t, s_t, a_t)$.

The encoder and the observation model (i.e. decoder) correspond to Gaussian distributions with mean and variance parameterised by a convolutional and a deconvolutional neural network respectively. The transition and the reward models are also Gaussian distributions with their mean parameterised by fully connected feed-forward neural networks. Their scale is given by an identity covariance and unit variance respectively. A more detailed description of the architecture is found in Appendix G.

Algorithm 5: SSM Agent

Input: Transition model, $p(s_t|h_{t-1}, s_{t-1}, a_{t-1})$, observation model $p(o_t|h_t, s_t)$, reward model $p(r_t|h_t, s_t, a_t)$, encoder q

Initialise: Replay buffer D, parameters θ, latent z_t, h_t, s_t

for $t \dots T$ **do**
- Compute $h_t = f(h_{t-1}, s_{t-1}, a_{t-1})$;
- Construct and sample $s_t \sim q(s_t|h_t, o_t)$;
- $z_{t:T} = [h_t, s_t]$;
- Act $a_t \sim \text{RHE}(z_t, p(s_t|s_{t-1}, s_{t-1}, a_{t-1}), p(r_t|h_t, s_t, a_t)))$;
- Observe $o_{t+1}, r_t \sim p(s_{t+1}, r_t|s_t, a_t)$;
- Append $D \leftarrow D \cup \{s', a, r\}$;
- **if** *done* **then**
  - $env.reset()$;
- **end**
- **for** $i \dots Epochs$ **do**
  - Sample trajectories $o_{t:T}, a_{t-1:T-1}, r_{t:T} \sim D$;
  - Compute $h_{t:T} = f(h_{t-1:T-1}, s_{t-1:T-1}, a_{t-1:T-1})$;
  - Construct and sample $s_{t:T} \sim p(s_{t:T}|h_{t:T})$;
  - $z_{t:T} = [h_{t:T}, s_{t:T}]$;
  - Construct $p(o_{t:T}|s_{t:T}, h_{t:T})$;
  - Construct $p(r_{t:T}|s_{t:T}, h_{t:T})$;
  - Update model by computing loss $\mathcal{L}(\theta)$ from equation 4.2;
- **end**

end

4.4 Planning with State Space Models

We now describe our main contribution. For the planning mechanics, we integrate the RSSM with RHE to generate, evaluate and select action sequences that are predicted to be the most favourable. The planner uses the forward model even during its earliest learning stages (before it has reached any level of accuracy) which establishes a bidirectional relationship. The planner is contingent on the reliability of the forward model to evaluate potential plans and the RSSM depends on the observations that are gathered by the actions mandated by the planner to improve itself.

Let us be more concrete about how RHE works in conjunction with the RSSM. RHE starts by either generating an action sequence $(a_t, ..., a_{t+H})$ of length H or reusing a previously generated sequence and applying the *shift-buffer* technique by shifting the sequence one step to the left and appending a new random action at the end of it. Here we assume that all actions are sampled uniformly. The sequence is used as the seed from where other $N - 1$ action sequences are generated via mutation. So far, this is the same familiar RHE process used in the other chapter, however, once RHE has generated the N candidate it requires a starting observation or state to execute the action sequences. At this stage, it is where the RSSM generates a latent state $z_t = \{s_t, h_t\}$ which is then cloned N times, one per each candidate sequence. The copies of the latent state and the action sequences are then passed on to the RSSM machinery which starts simulating in the compact latent space N trajectories in parallel according to the action plans, that is $h_{t+1:H+1} = f(h_{t:t+H}, s_{t:t+H}, a_{t:t+H})$ and $s_{t+1:H+1} \sim p(s_{t+1:H}|h_{t+1:H})$ to concatenate them as z. The RSSM gathers these latent state trajectories $(z_t, ...z_{t+H+1})_n$ which then can be evaluated by RHE according to one or multiple criteria. For the agents considered in this chapter, RHE selects the plan with the highest predicted return by invoking the SSM reward model $r_{t+1:H+1} \sim p(r_{t+1:H+1}|h_{t:H}, s_{t:H}, a_{t:H})$ and getting the cumulative sum of rewards. The RHE

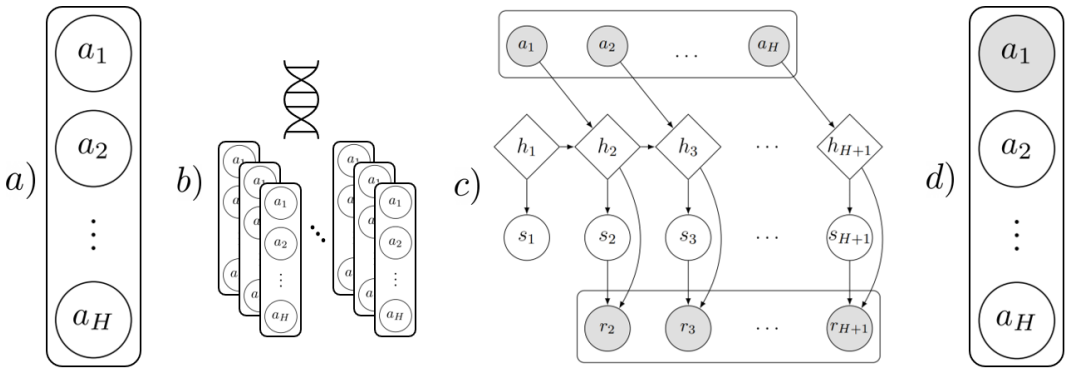


Figure 4.5: *The integration of a recurrent state space model (RSSM) with Rolling Horizon Evolution for online planning. a) The process starts by sampling a seed action sequence. b) N candidate action sequences are obtained from the application of genetic operators to the original action sequence. c) Each of the action sequences is used within the RSSM to simulate hypothetical trajectories and gather their corresponding rewards to evaluate them. d) From the top-ranked action sequence, we take the first action a_1 which is then executed by the agent in its actual environment.*

objective can be expressed as

$$\pi(h, s) = \underset{a_{t:t+H} \in \mathcal{P}}{\operatorname{argmax}} \mathbb{E}\Big[r(h_t, s_t, a_t) + \sum_{i=t+1}^{H} r(\hat{h}_{t+i}, \hat{s}_{t+i}, a_{t+i})\Big] \tag{4.3}$$

Although at the level of the implementation the RSSM does not make a distinction between actual and simulated latent states, we make this distinction explicitly in the expression for clarity. We can observe that for the first step the reward is evaluated on the actual latent state $z_t = \{h_t, s_t\}$ and on the simulated latent states $\hat{z}_t = \{\hat{h}_t, \hat{s}_t\}$ for the rest of trajectory. Fig. 4.5 graphically summarises the general idea behind the integration process between RHE and the RSSM.

Algorithm 6: RHE-SSM planner

Input: Latent h_t, s_t, Transition model, $p(s_t|h_{t-1}, s_{t-1}, a_{t-1})$, reward model $p(r_t|h_t, s_t, a_t)$, number of actions A, sequence length $\mathcal{T}$, population size $\mathcal{P}$, mutation rate μ

Output: Best action sequence π_{best}

Initialise: Matrix of action sequences π, last used π_{best}

if *not shift buffer* **then**
  $\pi \leftarrow a_t, \ldots, a_{\mathcal{T}} \sim Cat(A)$;
else
  Shift last used π_{best} to the left and add $a_{\mathcal{T}} \sim Cat(A)$ at the end of the sequence;
end
Tile π_{best} in π;
$\pi \leftarrow$ Mutate π with rate μ;
`/* Simulate trajectory from` h_t, s_t `*/`
Compute $h_{t+1:T+1} = f(h_t, s_t, \pi)$;
Construct and sample $s_{t+1:T+1} \sim p(s_{t+1:T}|h_{t+1:T})$;
Sample $r_{t:T+1} \sim p(r_{t:T+1}|h_{t:T+1}, s_{t:T+1}, \pi)$;
Evaluate sequences $R(\pi) = \sum_t^{T+1} r_t$;
Select best action sequence $\pi_{best} \leftarrow max_\pi R(\pi)$;

4.5 Experiments

4.5.1 Environment

We test the ability of an agent to learn and plan with an approximate forward model. Fig. 4.6 shows the environment, a top-down grid-based navigation task designed in Griddly (Bamford et al., 2020). In the frame, the agent is represented by the mouse and its opponent by the spider. The grid contains two absorbing cells, the cheese which acts as the goal, and the hole as a trap. Both cells terminate the task but if the agent reaches the goal it receives a positive reward of +1, on the other hand, if the agent gets to the hole the reward is −1. The agent also receives a reward of −1 if it comes into contact with the spider and the task terminates. The task has a time limit of 500 time steps if it is reached the task terminates and the agent gets a reward of 0. We conduct four variations of

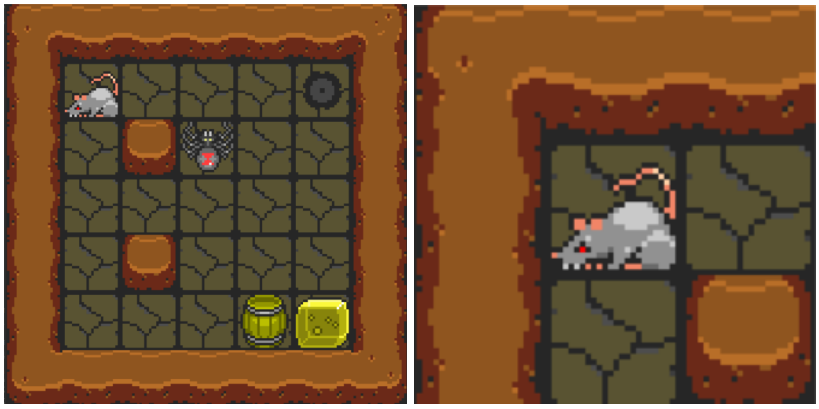


Figure 4.6: *The environment. There are two observation modalities. On the left, the observations gathered by an agent cover the whole grid and correspond to* 7×7 *cells (*168×168 *RGB pixels). On the right, the agent only gets a local view of its immediate* 3×3 *cell neighbourhood (*28×28 *RGB pixels).*

the experiment to analyse the performance of the agent along two important dimensions that emphasise aspects of partial observability. The first of them is an element of stochasticity provided by the adversarial spider which moves randomly throughout the grid. The other limits the observations that the agent has access to. In this case, whether the agent receives global observations that cover the whole grid or if they are only about their immediate surroundings. Thus the agent needs to construct a model that integrates and preserves information across time steps and that allows it to act under uncertain situations.

The environment is a 7×7 grid that corresponds to 168×168 RGB pixels. For the tasks where the agent has visual access to the whole grid, it receives a rescaled raw pixel observation of size $3 \times 64 \times 64$ down from the original $3 \times 168 \times 168$. If the agent only has a partial view of the task it receives observations of size $3 \times 28 \times 28$ which correspond to its surrounding 3×3 cell neighbourhood (fig. 4.6 right).

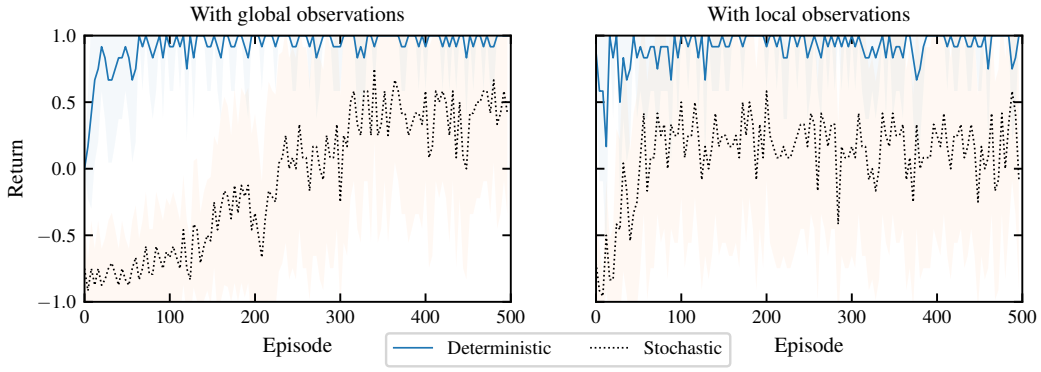


Figure 4.7: *The performance in the task over 500 episodes. The plots show mean and standard deviation over three seeds for deterministic and stochastic variations of the environment. Left: Global observations. Right: Local observations.*

4.5.2 Results

Deterministic variations. First, we describe the results in the deterministic versions of the task where the random opponent is not included. The performance of the agent is similar regardless of whether the agent had a global view of the environment or only of its local neighbourhood (fig. 4.7). Nonetheless, there are some differences concerning the number of steps it takes the agent to achieve the goal. It was noticed that when the agent constructs a forward model only based on local observations it is more efficient as it reaches the goal in fewer steps. As training progresses the agent with a global view of the environment starts reducing the number of steps to reach the goal and they become comparable. To analyse the behaviour of the agent from a qualitative perspective we also recorded videos tracking its interaction with the environment, and the simulated reconstruction given its current world model. In fig. 4.8 for example, it shows on the left frame the ground truth and on the right, the reconstruction extracted from the agent's forward model. We can appreciate that although the general visual structure of the task in the reconstructed frame matches the ground truth, it does not recreate the current location of the agent. This was a common theme early on during training, the agent was able to solve the task before it could learn

Figure 4.8: *Examples of ground truth (left column) and reconstructed (right column) frames. Top: The agent can solve the task before an accurate visual reconstruction has been learned. Middle: Another example of the agent performing and modelling the task. In this case, the spider acts as a stochastic element in the environment. Bottom: An agent's reconstruction of its local neighbourhood.*

to construct a model accurately.

Stochastic variations. Unlike in the deterministic version of the task, there were some palpable differences in the performance trends when the environment included elements of stochasticity. As we can observe in fig. 5.3 the agent exhibited a higher initial responsiveness if it only had access to local observations. Before it reached 100 episodes the agent had achieved a performance that was equivalent to an agent that had been trained for 300 episodes with global observations. Visualising again the episode recordings of the agents it was possible

to observe some behavioural patterns that suggested some interpretations of the results. For instance, when the agent collects global observations it was noticed that during the earlier stages of training it tends to favour plans that either keep it close to its starting position or that move in cyclical vertical patterns that maintain it behind the blocks. One could speculate then that the first interactions with the environment influenced the agent to develop a preference for conservative plans that could help it to avoid its adversary. Given that this occurs when the agent is just starting to construct a world model and that the opponent does not remain in a fixed position, the agent does not possess reliable assessments of where the opponent might be and those plans maximise the likelihood of not receiving a negative reward. However, even if the dynamics of the opponent are stochastic they still obey physical constraints, and thus as training progresses we observed that the agent learns to reconstruct the potential location of the opponent with a degree of reliability. Even when the predictions of the model about the opponent's moves do not match, the agent can recalibrate it by gathering new evidence. It is conceivable to suppose that once the forward model has learned to capture more of the potential effects of moving in other directions that the agent is now in a position to learn how to explore further and eventually reach the goal. We also speculate that when the agent only receives observations of its immediate neighbourhood it does not have to account for the dynamics of the opponent as it is not present in most of its observations. It could be that because of this reason it can operate from the beginning with a larger variety of plans as most of them do not necessarily lead to negative outcomes. Despite this as the results show when the agent gathers global observations it achieves a slightly better performance in the long term. Given the experimental setup presented here, a potential explanation for this is that even though learning a global forward model is more demanding it also affords more flexibility as it has learned to capture more complex dynamics. Meanwhile, an agent with a local forward model might not have necessarily learned to deal with all the potential

contingencies.

4.6 Discussion

In this chapter, we have adopted a different strategy to acquire and plan with an approximate forward model. We leveraged recent approaches in model-based reinforcement learning that frame the problem as state space modelling. We have extended such approaches by bridging the gap that exists in the integration between RL and consolidated online planning algorithms in discrete action spaces, such as RHE. Given that at the core of an SSM is an inherent temporal structure, it provides a formalism to deal with dynamic time series such as the sequential observations and actions of an agent (1) interacting with an environment or (2) simulating this interaction. Modelling the problem in this manner confers us with several advantages. For learning and inference, it enables efficient use of the rich stream of temporal information that exists in a sequence which can result in more accurate state estimation. An agent with a higher capacity to learn from temporal correlations could infer more reliably the current configuration of a system such as its location in space or its relation to other entities. For planning, if the agent has been trained with an architecture that explicitly replicates a sequential dynamical system it might simulate more consistent and coherent predictions across time.

The other central tenet of an SSM is the direct separation of observed and unobserved variables. The unobserved, hidden, or latent variables admit different interpretations. For world modelling in RL, a natural interpretation emerges if we consider that to summarise an image many of the details contained within it are redundant or not strictly necessary. Moreover, this interpretation also extends temporally to sequences as the latent variables may capture abstract notions of the effects of actions in the environment. For instance, as we noted in the experiments, an agent could learn representations that seemed to prioritise

the encoding of relations that would allow it to get to or avoid certain locations on the grid depending on whether they were favourable or unfavourable, but this occurred before the representation could capture it at a discernible graphical level (fig. 4.8 top). That is, the agent did not have to wait to learn to reconstruct itself to be able to navigate reliably toward a goal. These findings will become more significant in chapter 6.

The explicit decomposition of the space into observed and latent variables can also offer other benefits that extend from the conceptual into the practical realm. Online planning algorithms such as RHE are constrained by time and computational resources. They depend on the search and evaluation of a large number of plans to find an appropriate action sequence, the fact that the simulation of trajectories occurs in a more compact latent state is significant, because it allows the evaluation of larger or longer action sequences. Another practical aspect that is not exclusive to SSMs but that it certainly favours is its modularity. When we invoke the SSM to generate sequences of latent states, these sequences can be reused to derive multiple evaluation criteria (e.g. auxiliary tasks, intrinsic signals). We do not need to know beforehand what we are interested in predicting because these predictions remain independent of the beliefs produced by the SSM.

For the story we are developing in the thesis, it is also worth to point out other significant differences with respect to the Bootstrapped Transition Functions (BTF) architectures from the previous chapter. While in both, we model probabilistically the state transitions, there are important distinctions in the way this is approached. BTFs are essentially deterministic. However, by training multiple heads we can treat each prediction *as if* they were samples from an underlying distribution of neural network parameters. In the case of the SSMs the transition distribution is modelled explicitly. That is, the neural network learns the parameters that specify this transition distribution, from which then we can construct the distribution and produce N number of state samples.

The tasks in this chapter provide a reward signal only when the episode terminates, either because the agent reaches the goal or because it fails the task by falling into the trap or by getting caught by the adversarial element. From the results we obtained, this sparsity was not an issue because the grid has a relatively small size and does not require exploration strategies or more sophisticated planning evaluations. Ranking plans according to their predictive return was enough to achieve a good performance. However as we continue to examine the architectures in more complex scenarios such as those with larger grids or more adversarial elements, we will assess whether this remains a valid evaluation criterion, and if not, what other criteria we could use.

5 Planning with Models of Value Functions

5.1 Introduction

In the past chapter, the behaviour of the agents that we considered was guided by the predicted return that could be obtained along a hypothetical trajectory. For that effect, Rolling Horizon Evolution (RHE) in conjunction with the Recurrent State Space Model (RSSM) evaluated the quality of the action sequences that it had generated by assigning them a score dependent on the latent states that it traverses. The scores consisted of the cumulative predicted rewards. An issue that arises with this approach is that the evaluation is strictly limited by the length of a sequence. Consider for example the task from that chapter. It is essentially a task with sparse rewards where an agent only observes them if it reaches an absorbing state. Although this might not be an issue in a small grid let us imagine a larger grid. If the sequences generated by the planner are not as well directed or long enough to contain a path that reaches one of those absorbing states then two paths could be evaluated and treated as equivalent. This even if a state with an associated high or low reward follows immediately after these paths end. There are other different instances where an evaluation only based on the return might be problematic. Consider now the case of a non-sparse task such as Minipacman. An action sequence that leads the agent toward a string of edible cells, but with an incoming ghost, could be deemed more favourably than another plan that directs the agent through emptier corridors if the simulated trajectory is truncated before

the hypothetical encounter with the ghost. Thus preferably we want our planner to be able to assign scores not only based on the predictions of what could occur when an agent follows a trajectory but also an estimate of what could come beyond.

From optimal control and Reinforcement Learning (RL), the Bellman equation can recursively compute the value of a state, decomposing it into its expected immediate utility and the value of the next state. If an agent establishes these recursive relations then it is possible to form an estimate of what could occur when it reaches a state. Previously mentioned research (Hafner et al., 2020; Lee et al., 2020; Hafner et al., 2022), adopted an strategy reminiscent of classic model-based RL approaches such as Dyna-Q (Sutton, 1990, 1991). The agent uses the learned model to generate training sequences and learn to compute value functions and policy estimates. During execution time then the agent acts according to those estimates.

Here we take a different perspective to the problem of the estimation of the value functions. The main insight we propose is to treat the estimates produced by a value function as sequential time series in the same way we have done in the previous chapter with the observations, actions, rewards or states. In principle, if we can integrate them as part of a State Space Model (SSM) then we could produce a prediction on-demand of this estimate and simulate it along a trajectory. This is in contrast to the referenced architectures where the models are only used to compute more robust value and policy estimates but not to simulate plans during the actual decision-making stage.

Our main objective will be then to determine whether value estimation could mitigate some of the limitations imposed by plans evaluated exclusively on rewards. Our approach proposes (1) to learn a forward model of the value function within the SSM, and (2) provide RHE with the possibility to evaluate plans with rewards or Q-value estimates.

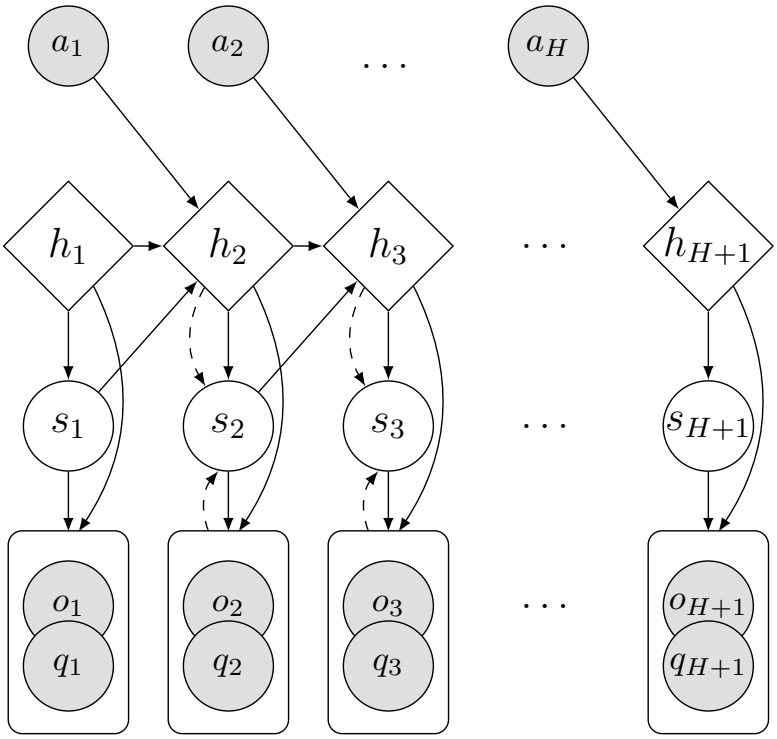


Figure 5.1: *Structure of the state space model with Q-values as observations. At each time step, the agent estimates potential Q-values which then can be treated as if they were also observations. The SSM learns the Q-values and can simulate them during planning.*

5.2 Models of State Value Functions

We take as a starting point the general RSSM structure described in section 4.2.4, in which the latent state is supported by a stochastic and a deterministic component. Each time the agent emits an action and transitions to a new state it gathers two measurements, the visual observation, and the reward. To model the dynamics we learn the transition model $p(z_{t+1}|z_t, a_t)$ and the observation model $p(o_t|z_t)$. The observation model, as before, serves as a target model and provides a self-supervised signal by comparing the predictions against the actual observations. Then we can make the following choice. We can try to learn a reward model $p(r_{t+1}|z_t, a_t)$ similar to the previous chapter and learn state-value estimates, or instead, we can avoid modelling the reward dynamics altogether. In that case, the reward is only used as an ephemeral piece of information to

form a subjective estimate of the value of a state-action pair. Accordingly, for this second RSSM variation we have the following components:

- Encoder $e_\phi(s_t|h_t, o_t, a_t)$
- Deterministic transition $h_t = \text{GRU}_\theta(h_{t-1}, s_{t-1}, a_{t-1})$
- Stochastic transition $p_\theta(s_t|h_t)$
- Observation model $p_\lambda(o_t|h_t, s_t)$
- Q-value model $p_\xi(q_t|h_t, s_t, a_t)$

We model the dynamics of the Q-values to estimate how beneficial or detrimental would be to emit action a_t in state z_t. Other estimates are also possible, for instance, another option is to model the state value function $V(z_t)$. However, we found that specifying explicitly the action provided more robust and useful estimates for planning. Also note that we have slightly modified the notation of the encoder from $q_\phi(s_t|h_t, o_t, a_t)$ to $e_\phi(s_t|h_t, o_t, a_t)$ to avoid confusion with a specific Q-value q. Figure 5.1 illustrates graphically the structure of this SSM. For more details about the architecture and the hyperparameter specification we refer the reader to Appendix G.

5.2.1 State Value Estimation

To learn to quantify the Q-values of state-action pairs we first start by defining two Gaussian distributions to act as the Q estimators, with their mean parameterised by fully connected feed-forward neural networks and unit variance. As it is common in the literature, one of these estimators is used to calculate the target values and is only modified gradually via soft updates (i.e. Polyak update). To obtain the targets $q^{\mathbb{T}}$ we sample chunks of trajectories from the buffer $(z_t, a_t, r_{t+1}, ..., z_{T+1}, a_{T+1}, r_{T+2})_n \sim B$ and compute the following expression:

$$q^{\mathbb{T}}(z_{t:T}, a_{t:T}) = r_{t+1:T+1} + \gamma \operatorname*{argmax}_{a_{t+1:T+1}} q^{\mathbb{T}}(z_{t+1:T+1}, .) \tag{5.1}$$

We also obtain the parameters of the predictive Q-value distribution $p_\xi(Q|z_{t:T}, a_{t:T})$ and then as we need to fit it to the target Q-values we maximise the log-likelihood,

$$\max_{\xi} \ln p_\xi(q^{\mathbb{T}}_{t:T}|z_{t:T}, a_{t:T}) \tag{5.2}$$

Algorithm 7: QSSM Agent

Input: Transition model, $p(s_t|h_{t-1}, s_{t-1}, a_{t-1})$, observation model $p(o_t|h_t, s_t)$, reward model $p(r_t|h_t, s_t, a_t)$, q-value model $p(q_t|h_t, s_t, a_t)$, encoder q
Initialise: Replay buffer D, parameters θ, latent z_t, h_t, s_t

for $t \dots T$ **do**
  Compute $h_t = f(h_{t-1}, s_{t-1}, a_{t-1})$;
  Construct and sample $s_t \sim q(s_t|h_t, o_t)$;
  $z_{t:T} = [h_t, s_t]$;
  Act $a_t \sim \text{RHE}(z_t, p(s_t|s_{t-1}, s_{t-1}, a_{t-1}), p(r_t|h_t, s_t, a_t), p(q_t|h_t, s_t, a_t)))$;
  Observe $o_{t+1}, r_t \sim p(s_{t+1}, r_t|s_t, a_t)$;
  Append $D \leftarrow D \cup \{s', a, r\}$;
  if *done* **then**
    $env.reset()$;
  end
  for $i \dots Epochs$ **do**
    Sample trajectories $o_{t:T}, a_{t-1:T-1}, r_{t:T} \sim D$;
    Compute $h_{t:T} = f(h_{t-1:T-1}, s_{t-1:T-1}, a_{t-1:T-1})$;
    Construct and sample $s_{t:T} \sim p(s_{t:T}|h_{t:T})$;
    $z_{t:T} = [h_{t:T}, s_{t:T}]$;
    Construct $p(o_{t:T}|s_{t:T}, h_{t:T})$;
    Construct $p(r_{t:T}|s_{t:T}, h_{t:T})$;
    Update perceptual model by computing loss $\mathcal{L}(\theta)$ from equation 4.2;
    Construct $p(q_{t:T-1}|s_{t:T-1}, h_{t:T-1}, a_{t:T-1})$;
    Construct and sample $q_{t+1:T} \sim p(q_{t+1:T}|s_{t+1:T}, h_{t+1:T})$;
    Compute $q(z_{t:T-1}, a_{t:T-1}) = r_{t+1:T} + \gamma \arg\max_a q(z_{t+1:T}, a_{t+1:T})$;
    Update q-value model by computing loss $\mathcal{L}(\theta)$ from equation 5.2;
  end
end

5.3 Plan evaluation with State Value Functions

The integration between the SSM and RHE as a planning algorithm follows the same basic mechanics as those described in section 4.4. RHE generates an initial action sequence $(a_t, ..., a_{t+H})$ of size H which is mutated to generate other $N-1$ sequences. Then, as exemplified in fig. 4.5, RHE invokes the SSM to simulate N rollouts starting from the current latent state $z_t = \{h_t, s_t\}$ using each of the different action sequences to sample $h_{t+1:H+1} = f(h_{t:t+H}, s_{t:t+H}, a_{t:t+H})$ and $s_{t+1:H+1} \sim p(s_{t+1:H}|h_{t+1:H})$. The crucial difference is that once we have gathered the rollouts we can call the Q-value model to sample values associated with the latent state-action pairs. We briefly advanced that we have a choice on how to use the Q-value estimates once we have learned their dynamics. We can follow a more conventional approach and estimate the n-step return on the simulated latent trajectory. This implies predicting the rewards up to the last latent state and then sampling a Q-value for the last latent state to account for the expected rewards beyond the horizon. The RHE objective can then be expressed as,

$$\pi(h, s) = \underset{a_{t:t+H} \in \mathcal{P}}{\operatorname{argmax}} \mathbb{E}\Big[r(h_t, s_t, a_t) + \Big(\sum_{i=t+1}^{H-1} r(\hat{h}_i, \hat{s}_i, a_i)\Big) + q(\hat{h}_H, \hat{s}_H, a_H)\Big] \tag{5.3}$$

where $z_t = \{h_t, s_t\}$ are the actual starting latent states and $\hat{z}_t = \{\hat{h}_{t+i}, \hat{s}_{t+i}\}$ are the simulated latent states in the rollout.

For the second SSM variation, we do not model the reward dynamics. Instead, the RHE policy π samples the Q-values for all steps of the trajectory $q_{t:H} \sim p(q_{t:H}|h_{t:H}, s_{t:H}, a_{t:H})$. The RHE objective can then be rewritten as,

$$\pi(h, s) = \underset{a_{t:t+H} \in \mathcal{P}}{\operatorname{argmax}} \mathbb{E}\Big[q(h_t, s_t, a_t) + \sum_{i=t+1}^{H} q(\hat{h}_{t+i}, \hat{s}_{t+i}, a_{t+i})\Big] \tag{5.4}$$

We are going to refer to the first objective as QSSM-RQ and to the second objective as QSSM-Q. We will see in the experimental results that there might

be circumstances where it can be advantageous to choose one objective over the other. The state-action value is interesting in itself, despite that it can be vastly inaccurate and suffers from overestimation it is a much richer signal than the reward. It is an aggregate that encapsulates information about the possible paths that can be taken from a state. Thus we could treat it as if it was a *potential* function for the vector field comprised of all latent state-action pairs.

There is another interpretation that we could give to this QSSM-Q objective. Since RHE is an open-loop planner, it gives rise to what are essentially uncontrolled processes where actions are randomly chosen and disconnected from each other. Given this independence, by estimating the Q-value for each state-action pair we can accordingly treat each step as a separate decision-making problem, and as if the quality of the decision was being analysed by an external observer.

Algorithm 8: RHE-QSSM planner

Input: Latent h_t, s_t, Transition model, $p(s_t|h_{t-1}, s_{t-1}, a_{t-1})$, reward model $p(r_t|h_t, s_t, a_t)$, q-value model $p(q_t|h_t, s_t, a_t)$, number of actions A, sequence length $\mathcal{T}$, population size $\mathcal{P}$, mutation rate μ

Output: Best action sequence π_{best}

Initialise: Matrix of action sequences π, last used π_{best}

if *not shift buffer* **then**
  $\pi \leftarrow a_t, \ldots, a_{\mathcal{T}} \sim Cat(A)$;
else
  Shift last used π_{best} to the left and add $a_{\mathcal{T}} \sim Cat(A)$ at the end of the sequence;
end
Tile π_{best} in π;
$\pi \leftarrow$ Mutate π with rate μ;
`/* Simulate trajectory from` h_t, s_t `*/`
Compute $h_{t+1:T+1} = f(h_t, s_t, \pi)$;
Construct and sample $s_{t+1:T+1} \sim p(s_{t+1:T}|h_{t+1:T})$;
if *evaluate with rewards* **then**
  Sample $r_{t:T+1} \sim p(r_{t:T+1}|h_{t:T+1}, s_{t:T+1}, \pi)$;
  Evaluate sequences $R(\pi) = \sum_t^{T+1} r_t$;
else if *evaluate with q-values* **then**
  Sample $q_{t:T+1} \sim p(q_{t:T+1}|h_{t:T+1}, s_{t:T+1}, \pi)$;
  Evaluate sequences $R(\pi) = \sum_t^{T+1} q_t$;
else
  Sample $r_{t:T} \sim p(r_{t:T}|h_{t:T}, s_{t:T}, \pi)$;
  Sample $q_{T+1} \sim p(q_{T+1}|h_{T+1}, s_{T+1}, \pi)$;
  Evaluate sequences $R(\pi) = \sum_t^{T} r_t + q_{T+1}$;
end
Select best action sequence $\pi_{best} \leftarrow max_\pi R(\pi)$;

5.4 Experiments

5.4.1 Environment

We compare the performance between agents planning with Q-value models and those with reward models in two environments. The first is Minipacman as presented in chapter 3. The second is based on the classic four-room environment and will be described below. We create a wrapper for Minipacman to pass to the agent the raw pixel observations instead of cell categories. This implies that the agent receives $3 \times 64 \times 64$ observations instead of the $3 \times 15 \times 19$ observations that the error-correcting agents from chapter 3 receive. We keep the same rules and rewards that were also described in that chapter.

For the second environment, we design a 13×13 cell grid based on the classic four-room environment. For the moment we only consider our agents and in the next chapter, we will extend our analysis to compare them against Dreamer v2 (Hafner et al., 2020) and MuZero (Schrittwieser et al., 2020). There are four different variations of the task that we will consider:

- Standard four-room: this is the classic formulation where an agent (mouse) is located in one of the rooms and must attempt to navigate to a different room to reach a goal indicated by the cheese. If the agent reaches the goal it receives a reward of +1.
- Stochastic adversary four-room: in this variation, we add an adversarial element (green snake) located initially in one of the other rooms, as shown in fig. 5.2b. This element moves randomly through the grid. If the agent encounters it, it receives a reward of −1.
- A* adversary four-room: similar to the previous variation, this environment includes an adversarial element (spider in fig. 5.2c). Instead of moving randomly, the adversarial element finds the shortest path via the A* search algorithm to try to intercept the agent.

- Two adversary four-room: the last and most challenging variation includes the two adversarial elements (fig. 5.2d).

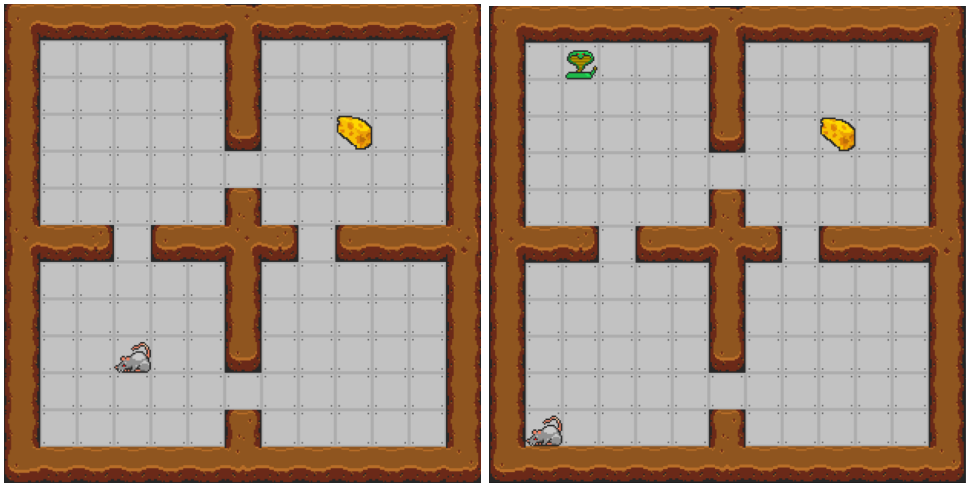
The grid corresponds to a raw pixel observation of size $3 \times 312 \times 312$ rescaled to $3 \times 64 \times 64$.

5.4.2 Results

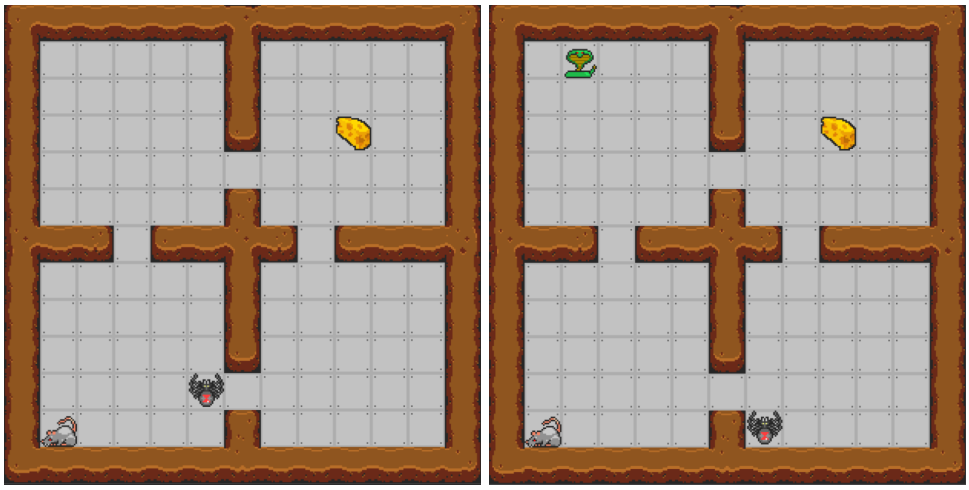
Four-Room tasks

We tested three different agents in each of the tasks. The first of those is the agent from the previous chapter which evaluates the candidate plans according to a sum of rewards along the trajectory. We will refer to this agent as SSM-R. The second agent (QSSM-RQ), takes the sum of the rewards in a sequence up to the last step, where it obtains a Q-value to account for the expected future sum of rewards beyond the planning horizon. As it was mentioned earlier, it can be considered that this a more common way to calculate the expected return. The third agent (QSSM-Q) computes a simulated Q-value at each step, gets their sum uses it to evaluate the sequence.

As can be observed in fig. 5.3 (top left), all agents become proficient without major issues on the first level. The QSSM-Q agent takes slightly more episodes to fully become proficient in the task. This might be because, in contrast to the rewards, the Q-values are more challenging to learn and require stabilisation techniques such as the use of a Q-target network. For the second level, we start to appreciate the advantage that computing an estimate could have for an agent. The two agents that employ simulated Q-values (QSSM-Q and QSSM-RQ) start to consistently solve the task before they reach 400 episodes. Meanwhile, the SSM-R agent, using only simulated rewards to select its plan, drifts around 0 return. This is due to a combination of the three possible outcomes, either achieving the goal, running out of time, or getting intercepted by the stochastic agent. Because this agent has a weaker signal some of its interactions with the



(a) Level 1 *(b) Level 2*



(c) Level 3 *(d) Level 4*

Figure 5.2: *Four-Room environment levels. Level 1: Classic formulation. The agent (mouse) has to reach the goal located in one of the other rooms (cheese). Level 2: It includes an adversarial stochastic element (green snake) that moves randomly throughout the grid. If the agent collides with it, it receives a reward of −1. Level 3: In this level, the adversarial element moves purposely toward the agent, as it plans its moves via the $A^\star$ search algorithm. Level 4: Both adversarial elements are included in the grid.*

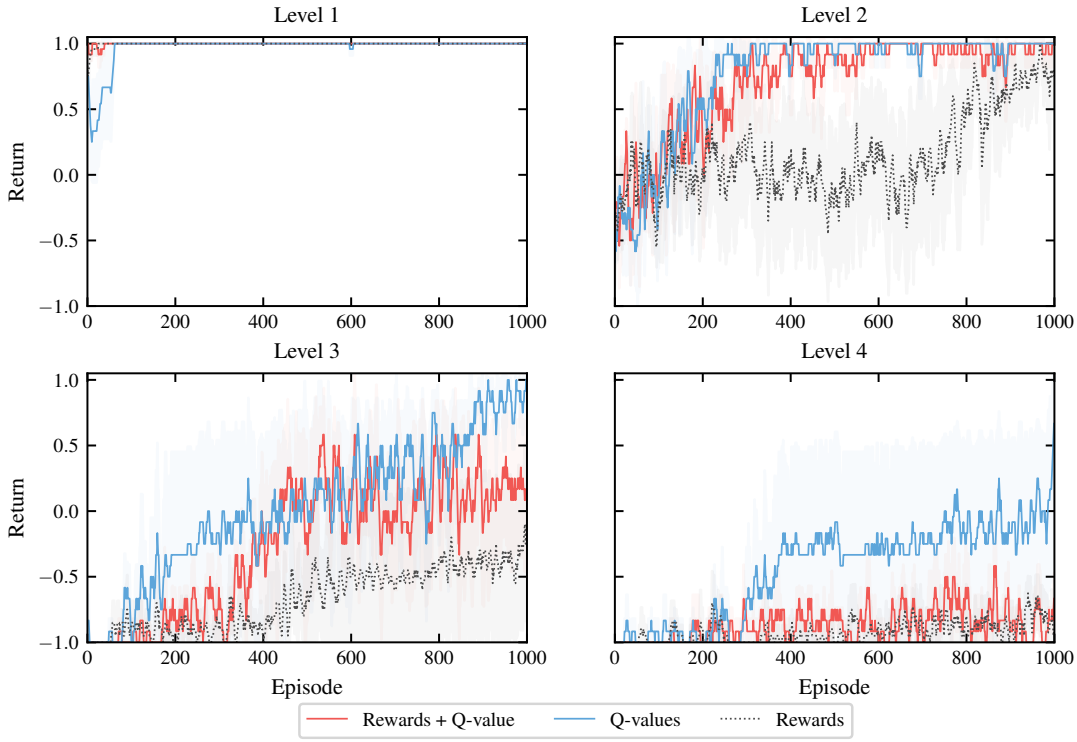


Figure 5.3: *The results show the performance of the agents in each level across 1000 episodes. We compare three different plan evaluation criteria: (1) sum of rewards along the trajectory (SSM-R), (2) Q-value in the last plan step plus rewards in the steps preceding it (QSSM-RQ), and (3) sum of Q-values along the trajectory (QSSM-Q).*

environment may lead to an inefficient exploration of the environment and it takes longer to gather enough evidence to facilitate learning where is the goal. The agent eventually starts to improve and increases its return, however, this occurs until the last 200 episodes.

For levels 3 and 4 that include the A* adversary, the differences between the QSSM-RQ agent and the other two become more significant. For instance in fig. 5.3 (bottom left) that corresponds to level 3, the Q-values agent exhibits a gradually increasing trend. By the final episodes, the agent has achieved proficiency in the task that is near the maximum on average. The QSSM-RQ agent also showed initial steady progress however it then stabilises and does not continue its progression. The decline in performance for all agents is more significant in level 4 (5.3 bottom right). The full SSM-R and the QSSM-RQ

agents exhibited major complications to get to the goal or even surviving within the specified time limit. By comparison, the QSSM-Q agent was able to do it more frequently and showed and maintained a slight positive tendency towards the end. Nonetheless, its performance was not consistent.

There was a noticeable qualitative behaviour that we observed when the agent was in the presence of the A* adversary. During the early training episodes, the agent learns first how to escape the adversary when it is detected within proximity. The agent generally stays by the walls and performs an abrupt side move in the same room where it is currently located and waits until it again detects that the adversarial element is approximating. If the wall connects to another room, then the agent moves to it and eventually leads to the discovery of the goal.

Minipacman

Figure 5.4 shows the results in Minipacman over 1500 episodes. As a reference, we have included in the plot an error-corrected BTF agent from chapter 3. It corresponds to an agent with a planning horizon for five-time steps. There are some differences in the schemes and in their training process that makes a one-to-one comparison difficult to establish. We can, however, discuss aspects that are positive about one approach or the other. For example, in the BTF agents, the model learning and the planning stages are separated. The agent first learns a forward model in 50,000 training steps before it is used for planning and error-correction. This is to increase its stability. In contrast, in the SSM agents, these processes are interleaved and do not need any pre-training stage. It is important to note that the non error-corrected BTF and BTF+RPF agents exhibit lower performance than the SSM agents. This demonstrates the benefits of using information that is generated by its ensemble to boost the accuracy of the predictions and the performance. Although this requires domain knowledge regarding the specific elements that we can find in the environment. Thus an obvious advantage of the SSM agents is their generality as they show competitive

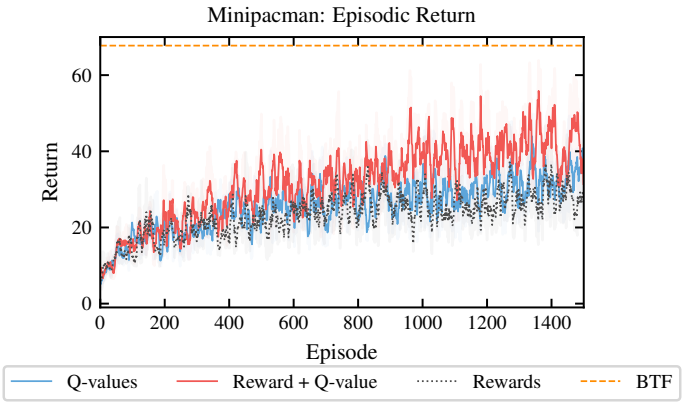


Figure 5.4: *Episodic return in Minipacman. The comparison shows the SSM agent from chapter 4 (SSM-R), and the two agents introduced in this chapter, Q-values (Q-values) and RSSM-RQ (Reward + Q-value). We also include as a reference the error-corrected BTF from chapter 3 that exhibited the best performance in this task.*

performance despite their larger observation space (i.e. $3\times15\times19$ vs. $3\times64\times64$). There was an important aspect we noticed when we analysed the videos to observe the behaviour of the SSM agents when they tried to solve the task. If the agent had already eaten the ghost and was in a part of the environment where it had also already eaten the pills on both sides of the corridor it seemed to be unable to go to sections where it had not. Without the pressure exerted by the ghost or any incentives from pills in its surroundings, the agent tended to move cyclically between a few cells. We illustrate an example of this situation in fig. 5.5. We concluded that this could occur because it becomes a sparse reward task. Without pills in its proximity, the closest pills are either beyond its planning horizon or what it can estimate with its current forward model. From this, we can also draw some lessons from the BTF agents. Because the BTF agents are ensemble-based each of its approximators generates its own, and potentially divergent, prediction. This mechanism grants these agents an inherent drive towards exploration, especially when the agent is in a region of the state space that it has not experienced often. Thus it is interesting to speculate if extending the SSM to an ensemble architecture could alleviate these issues that we observed.

If we focus exclusively on the SSM results we can also observe some differences in performance. Unlike the results we obtained in the Four-Room tasks, in Minipacman the QSSM-RQ agent, has the best performance. We can also observe that even when the agent only models the rewards (SSM-R) its performance is not hit as much as it is in some of the Four-Room tasks. A potential explanation for this might be due to the different reward structures in the environments. In the Four-Room tasks, the rewards are only encountered in absorbing states. The agent observes the reward when it reaches the goal or when the adversary intercepts it. In some episodes, it may not even encounter any reward if it exceeds the time limit. Given all these circumstances, a trained reward model is less informative and therefore less useful if almost every prediction is the same regardless of the plan. The Q-value estimates provide a much richer guiding signal. However, in Minipacman the rewards appear frequently and they have different values. Thus it is possible to rank the plans in a more meaningful manner than in the Four-Room tasks. In the particular comparison between the QSSM-RQ and the QSSM-Q agents, we speculate that the former exhibits a better performance because the rewards are learned over concrete and static targets unlike the estimates of the Q-values. Therefore when reward observations are frequent it could be a better strategy to rely on the reward estimates rather than on a Q-value which is itself an estimate of the expected reward.

Figure 5.5: *Scene of the Minipacman environment. The left frame shows the ground truth and the right frame its reconstruction.*

5.5 Discussion

In this chapter we have deviated from recent architectures such as Dreamer (Hafner et al., 2020, 2022) or SLAC (Lee et al., 2020) in the way to approach state-value estimation in a model-based setting. In those architectures the simulated trajectories generated by the approximate model are used to obtain a policy. When the agent interacts with the actual environment the model is not used to simulate trajectories and instead uses the learned policy. The SSM in those cases is used to facilitate the acquisition of more robust behaviours and representations. In contrast, in our model-based approach, the agent also uses the SSM to communicate with a planning algorithm to look ahead via the simulated rollouts. Rather than training a policy, RHE searches the policy space assisted by the reward and the Q-value models. This is also precisely another important difference. To accomplish this we can treat the Q-values as another observation that can be simulated given a current latent state and action. Thus we learn the parameters of the distribution that govern the Q-value transitions. To an extent, this is similar to MuZero (Schrittwieser et al., 2020) which predicts a state value and a policy during an Monte-Carlo Tree Search (MCTS) rollout. Here however we model the Q-values contextualised as part of an SSM and in a probabilistic

setting.

Following from the previous chapter, our objective was to generalise the architecture further by relaxing the dependency on the planning horizon. From dynamic programming and the Bellman equation, computing a state-value function provides an estimation of the value of its current and remaining steps in a sequence. In addition to propose an architecture to learn Q-value models as part of the SSM, we presented two schemes to integrate the Q-value model as part of the architecture. The first follows a more conventional approach where the reward model is used up to the last state of the simulated trajectory and then the Q-value model estimates the expected rewards beyond the planning horizon. For the second, we remove the reward model and the agent simulates exclusively Q-values for each step of the sequence. We noted that we could treat the action sequence as a series of independent decisions. For example, if we consider the results from the four-room tasks, we observed that if we take advantage of this assumption and simulate a Q-value for each step of the trajectory, the agent performs better than when the Q-value is considered only beyond the planning horizon. However, the results we obtained were inconclusive and were not replicated in Minipacman where the QSSM-RQ agent that combined rewards with Q-values performed better. We speculate that this could be due to the differences in the reward structures of the tasks. The four-room tasks are sparse and defined by absorbing states. In these situations, the Q-value can be a more informative signal for the planner, and evaluating them according to this criterion could improve its ability for selecting among competing trajectories with seemingly identical returns. In Minipacman rewards are obtained more frequently and thus the model can learn concrete reward estimations and their difference depending on the state. Thus evaluating the rollouts with rewards might produce less noisy estimates of the actual value of trajectory. We note however, that adding the Q-value at the end of the planning horizon proves to be crucial for increasing the performance of the agent compared to planning only with rewards. This

appears to lend support to the conjecture raised in this chapter regarding the shortcomings from planning only with rewards. Adding the predicted Q-value at the end of the trajectory provides an estimate of future rewards beyond the planning horizon.

Another salient point from the results obtained in Minipacman is the performance of the error-corrected Bootstrapped Transition Functions (BTF) architecture compared to the different SSMs. While it is true that the training regimes are dissimilar (i.e. in BTFs we first train the model before it can be used for planning, while in the SSMs learning and planning are intertwined), the difference in performance was significant. We speculate that the type of representations learned by the model may play a role. In the case of the BTFs there is more domain knowledge involved in training the model. For example, the model learns a grid of cells, where each cell represents a specific category or object. This confers the model with a basic notion of what objects are present in the task. Whereas in the SSMs the architectures learn from raw pixels without any notion of the structure of the environment. However, this also raises the question regarding how to improve the representations learned by the model? And what should these representations refer to?

6 | Planning in Latent State Spaces with Non-reconstructive Forward Models

6.1 Introduction

In this thesis we have considered the situation in which an agent, when presented with an input state and the opportunity to select an action, hypothesises about the future by predicting the next states. We have assumed that the world model the agent uses should be constructed based on being accurate about the future. Thus far, we considered a strict interpretation of this objective. The agent learned next step transitions by learning to predict the next observations either directly, or indirectly through a latent state, which then are compared against the actual observations. The crucial aspect of this process is that the target predictions belong to the same space which gives rise to the actual observations. There is a sensible justification for trying to proceed in this way. If a model-free agent makes a decision based on the current input then a model-based agent with the capacity to generate future potential inputs should, under ideal conditions, select appropriate actions as if they were acting on inputs generated by the environment. Nonetheless, as we have seen, those ideal conditions are hardly attained. The environment may have complex dynamics and stochasticity, or the architecture might not be expressive or powerful enough. This gives

rise to situations where an agent plans according to predicted inputs that may not resemble those that come from the environment and thus render the plans useless. While in chapter 3 we proposed error-correcting schemes to enhance the accuracy of the predictions, in chapters 4 and 5 we examined how we can relax these constraints and instead model directly the parameters of a distribution from where to sample the predictions. This step also involved the inclusion of latent variables, which alleviated the need to produce simulated observations during the rollouts. We defined the latent representations to exist in a smaller space compared to the pixel-space that defines all possible observations. This implied that the latent representations were more compact and therefore reduced the error surface because they compressed the original observation removing irrelevant details and focusing on essential aspects that can favour their reconstruction. Also, because the latent space can be considerably smaller than the observation space it could be speculated that a large number of similar observations might be mapped as neighbors to a small area in the latent space (Mnih et al., 2015). If an actual observation is transformed into its latent counterpart and it differs from the simulated representation that an agent was expecting to encounter, the difference between them could be small enough that it does not affect in a major way the behaviour of the agent.

Despite that the latent states in the Recurrent State Space Model (RSSM) are enough for planning, and that there is no necessity to produce a trajectory in pixel space or other raw spaces, it is important to recall the learning objective expressed in equation 4.2, which we restate for clarity,

$$\ln p(o_{1:T}|a_{1:T}) \geq ELBO = \sum_{t=1}^{T} \mathbb{E}_q[\ln p(o_t|s_t)] - \mathbb{E}_q[D_{KL}[q(s_t|o_{\leq t}, a_{<t})||p(s_t|s_{t-1}, a_{t-1})]] \tag{6.1}$$

We can observe two things. First, it explicitly specifies that the goal is to ap-

proximate $p(o_{1:T}|a_{1:T})$. That is, we still model the observations and how the observations occur given an action. Second, to approximate this distribution we see that the first term on the right-hand side is a reconstruction term. Indeed, let us remember that one of the components in the architecture was the observation model and that its role during learning was to provide a log score of how well a latent state s_t reconstructs an observation o_t. That is, we notice that the objectives we have been implementing are still based on the notion of recreating external observations.

Based on this, we could ask the question of what else

Beyond the technical obstacles to produce detailed forecasts of the dynamics of an environment, in other fields such as psychology or neuroscience, there have been studies suggesting that humans are not particularly reliable to recall specific details (Wells et al., 2006; Drivdahl and Hyman, 2014; Nash et al., 2017). Moreover, phenomena such attention processes may involve brain structures sensitive to salient events marking them for additional processing and potential control signals (Menon and Uddin, 2010), and for the detection of behaviourally relevant stimuli dependent on current goals, needs or objectives (Corbetta and Shulman, 2002). Some researchers have speculated that the role of perceptual and representation mechanisms is to be sensitive to aspects of the world in terms of what they afford to an agent and not necessarily to be descriptive about the world (Gibson, 1979; Cisek, 2007; Pezzulo, 2008).

While unpacking these viewpoints in terms of world modelling is beyond our current scope, and we certainly cannot contribute substantially to any debate, there are several important ideas that can guide us to construct alternative training objectives. If we assume that an agent filters out part of a signal to prioritise certain bits of information over others. The question is what makes these pieces of information different from the rest and how can they be identified? In principle, these questions may seem to be contextual. What is valuable or relevant

may depend on the agent's current objective. However, to start to tackle these questions we can take a broader perspective and relate the insights provided above to the original assumption about predicting the future. First, we start this chapter by reviewing some notions of relevance that have been formulated with information-theoretic principles. We then provide some justifications for why it may be a reasonable starting point from where to derive a new training objective for learning a forward model. The objective we propose, although predictive between present and future, does not make reference to the future observations. The concern ceases to be accurate about observations, and instead, the focus is on being predictive about latent factors, which we assume already capture essential features for the description of the environment. In addition, we also show how to make this objective practical. Using noise contrastive estimation techniques (van den Oord et al., 2019), we then approximate this objective to learn a model and perform online planning.

6.2 A brief detour into information bottlenecks

Let us start first by building some intuition about the notion of relevance [1]. Consider $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, where x may correspond to an input signal and y to a quantity of interest. We are interested in how much information x carries about y, or to put it in the language of information theory, the mutual information $I(X;Y)$. Here we assume that X and Y are not independent and therefore $I(X;Y) > 0$. For example, consider a scenario where a patient must be diagnosed with a health condition. The patient has to undergo several studies with the expectation that they are informative about her condition. If we considered all possible medical studies X then it is maximally informative to determine what is the medical

[1] Note that we start by referring generically to relevance as information that is connected to a concrete objective. This is not to be confused with the specific notion of *relevant information* (Polani et al., 2001, 2006) that has been defined in the context of the perception-action loop as the information necessary to achieve certain level of performance.

condition Y. However we can split the measure space $\mathfrak{X}$ into two subsets: $\mathfrak{U}$ that contains medical studies that are non-informative to determine the condition thus $I(U;Y) = 0$, and $\mathfrak{X}$ which on the other hand is informative about Y so $I(Z;Y) > 0$. That is, among all the possible medical studies X that a person could have, only a subset measured by Z offers information that is *relevant* to correctly determine what is the medical condition Y. Although we could opt for the larger space of events measured by X to guarantee that it will be maximally predictive about Y there are several reasons why this might not be an appropriate option. A larger data stream is constraining, it implies more processing costs, bandwidth, time investment, or storage capacities among other factors. Thus it could be preferable to aim to find Z with mutual information $I(Z;Y)$ as close as possible to $I(X;Y)$. The *information bottleneck* (Tishby et al., 2000) allows us to precisely formulate this trade-off as a variational principle

$$\min_{p(z|x)} [I(Z;X) - I(Z;Y)] \tag{6.2}$$

The expression above indicates in the first term that we must minimise the amount of information Z takes from X while simultaneously the second term tells us that Z must maximise the ability to inform about Y. The information bottleneck is framed as an optimisation problem where we must learn a mapping from x to z, here given by the conditional distribution. For simple problems, this mapping can be iteratively calculated until convergence via a generalisation of the Blahut-Arimoto algorithm (Tishby et al., 2000) and for more complex scenarios we can learn it with a neural network (Alemi et al., 2016).

6.2.1 Predictive information

Let us contextualise further the problem of extracting from a time series information relevant for completing a task. Assume an environment that emits a data point at each time step $x = (..., x_{t-T}, ..., x_t, x_{t+1}, ..., x_{t+T}, ...)$. A data point may

contain what the agent can sense (e.g. observation, reward), but also what is non-perceptible to the agent. We split the sequence into two sub-sequences for past $x_{past} = (..., x_{t-T}, ..., x_t)$ and future $x_{fut} = (x_{t+1}, ..., x_{t+T}, ...)$. Because the environment has structure and correlations in space and between past and future, x_{past} can be used to predict part of the future x_{fut}. The *predictive information* (Bialek and Tishby, 1999; Bialek et al., 2001)[2],

$$I_{pred} = I(X_{past}; X_{fut}) = H(X_{fut}) - H(X_{fut}|X_{past}) \tag{6.3}$$

here intuitively expressed as the reduction of entropy in the future if we know the past. The predictive information then quantifies to what extent the past predicts the future. The entropy of the generative process of the environment is an extensive quantity whereas the predictive information is sub-extensive. For this reason, predictive information has also been proposed as a measure of the complexity or the degree of structure of a process. As mentioned, we expect that an environment has correlations in time. However, they tend to be a small fraction compared to the full data stream of the process and in time they should become negligible[3],

$$\lim_{t\to\infty} \frac{\text{Predictive Information}}{\text{Total Information}} = \frac{I(X_{past}; X_{fut})}{H(X_{past})} \to 0 \tag{6.4}$$

6.2.2 Predictive information as a criterion for relevance

Predictive information has been applied in the context of the perception-action loop for the generation of complex behavioural patterns (Martius et al., 2013, 2014; Martius and Olbrich, 2015) and in model-free Reinforcement Learning (RL)

[2] also known as *excess entropy* (Crutchfield and Packard, 1983) or *effective measure complexity* (Grassberger, 1986)

[3] Another reason why the predictive information can be small is if the dynamics of the generative process are very regular.

to provide a state representation to a soft actor-critic agent (Lee et al., 2020). Here, let us take the predictive information thesis and analyse its implications for world modelling. While being able to extract the pieces of data that can inform about the future is imperative, only a small fraction of the data stream of the past is necessary for prediction. Because it might not be ideal to keep all of the information produced by a process, then under ideal conditions, an agent would store in its memory only a portion of past data that is *relevant* to predict the future. We can formulate precisely this trade-off with an information bottleneck (Still, 2014),

$$\min_{p(z|x_{past})} [I(Z;X_{past}) - I(Z;X_{fut})] \tag{6.5}$$

According to this expression then an agent has to attempt to learn a mapping of past data points X_{past} to a memory or representation Z. The agent must aim to capture as little as possible from the past while maximising its predictive capabilities.

Several arguments could justify this approach. From a learning theory angle, we can consider the past as the training set, the future as the testing set, and the representation as the parameters learned by an algorithm. During the training stage, we train an approximator to capture the regularities and patterns from the data, but ultimately our interest is not to learn the training set but to generalise to unseen data. This is captured by the second term, $\max I(Z;X_{fut})$. To prevent the approximator from overfitting the training data we could constrain its complexity using regularisation techniques. This would correspond to $\min I(Z;X_{past})$.

Information is of particular importance for biological agents, large amounts of energetic resources are invested in information processing. For instance, it has been estimated that 20% of a human's resting energy expenditure is used in the brain (Laughlin et al., 1998) while in blowflies 8% is used by their retinas (Niven, 2014). The huge costs are compensated by the capacity to acquire information

that enables more sophisticated behavioural strategies. However, also due to these high energetic costs a suboptimal information processing strategy would be highly detrimental to the organism. From an evolutionary perspective, there is a relationship between cost and performance but also a tendency to minimise processing costs while achieving the same behavioural efficiency (Polani, 2009).

This also has found support in theoretical analyses in non-equilibrium thermodynamics. In Still et al. (2012), the authors derive a relation between the predictive power of a system and its thermodynamic efficiency. The main result in the paper is the instantaneous non-predictive information I_{np},

$$\beta \mathbb{E}[W_{diss}] = I(Z_t; X_t) - I(Z_t; X_{t+1}) = I_{np} \tag{6.6}$$

the equation highlights that the (expected) quantity of energy lost by a system as heat (i.e. dissipated work W_{diss}) is equal to the amount of information that is irrelevant to predict the next step into the future. The more non-predictive information a system retains the larger the amount of energy lost by the system and the more thermodynamically inefficient it is.

6.3 A Non-Reconstructive Predictive Model

The principles delineated in the previous section have considered unperturbed dynamics in which a passive observer collects observations and forms representations. Let us take the previous arguments to propose motivate our approach for the acquisition of a model. The basic problem formulation follows from chapter 4. An agent senses observations o and rewards r, and interacts with the environment through action a at each time step. The agent learns a latent representation z which carries information from the past $\overleftarrow{z}$ and aggregates it with the information extracted from the present's sensory stream.

The first aspect to note from the state space models described before is that to approximate the defined target distribution $p(o_{1:T}|a_{1:T})$, the terms in the variational objective also approximate individually two mutual information. Therefore we can reinterpret it in terms of an information bottleneck trade-off,

$$\min_{q(z|o)} [I(\{O_t, Z_{t-1}, A_{t-1}\}; Z_t) - I(Z_t; O_t)] \tag{6.7}$$

Where the first term $I(\{O_t, Z_{t-1}, A_{t-1}\}; Z_t)$ minimises the information to extract from past and present. In principle, then this term encourages discarding information that has been stored in the representation at previous time steps, and it serves as a regulariser. The second term, $I(Z_t; O_t)$, is a reconstructive term and promotes that the latent state contains as many informative elements as possible to reproduce the current observation.

When we discussed the State Space Model (SSM) in chapter 4, we argued that the explicit association of the observations to a compressed latent state was a mechanism to account for the essential factors for the reconstruction of an input stream. The information-theoretic interpretation increases our intuition about it and conveys precisely that Z_t holds information that is relevant to determine the current observation O_t. This also allows us to notice some aspects that might not be desirable for constructing world models. First, it is not explicitly predictive of the future. Z_t might provide a context for the agent to act but it is not directly boosting its anticipatory capacities. We could opt for a naive approach based on the predictive information information bottleneck from expression 6.5 such as,

$$\min_{q(z|o)} [I(\{O_t, Z_{t-1}, A_{t-1}\}; Z_t) - I(Z_t; O_{t+1})] \tag{6.8}$$

however, this implies that we still consider that a forward model should aim to predict an observation, in this case, the next step future observation O_{t+1}. As we commented at the beginning of this chapter, we may want to try alternative

prediction targets that do not concentrate on an accurate reconstruction of an observation. Consider again the relation between a raw observation, a latent state, and an information bottleneck. The agent's sensors have access to the observations, which are then filtered out and encoded in a considerably smaller latent state. These same sensors will receive observations in the future which then again are encoded as latent states. Recall that in the previous SSMs one of the components consists of a transition model in latent space used to generate trajectories. This is also the level at which the agent is operating for decision-making, not at the level of raw observations. Thus we can go a step further and establish the next latent state Z_{t+1} as the relevance variable which results in the following information bottleneck,

$$\min_{q(z_{t:t+1}|o_{t:t+1})} [I(\{O_t, Z_{t-1}, A_{t-1}\}; Z_t) - I(\{Z_t, A_t\}; Z_{t+1})] \tag{6.9}$$

Note that although the encoder is extracting features from the current observations and past context, it might not be apparent what it is capturing from them. It does not have a well defined training signal in the form of actual observations to reconstruct. The first term remains unchanged as we still want to limit the amount of information that we retain from the past or take from the current observation. However, the second term encourages the representation to preserve information that can predict its future evolution in latent space.

6.4 Learning a Non-Reconstructive Model

The previous expression 6.9 that we have just proposed includes the minimisation and maximisation of two mutual information. This has significant technical implications that we must consider before we can implement it as a learning objective. The most important one is to determine how to address the issue of estimating and maximising the mutual information. In this section we derive a

practical learning objective that approximates the expression 6.9 and can then be used to learn a non-reconstructive model.

The estimation and optimisation of mutual information are challenging and active problems of research because computing the mutual information is intractable for all non-trivial cases. Previously, the estimation of the mutual information had been limited to problems of low dimensionality (Kraskov et al., 2004; Gao et al., 2015) but given its preponderance in many areas of machine learning, it has received attention in deep learning resulting in several advancements. For example, to maximise a lower bound the neural networks have been used as estimators based on dual representations of the KL divergence (Donsker and Varadhan, 1976; Nguyen et al., 2010; Belghazi et al., 2018), the Jensen-Shannon divergence (Hjelm et al., 2019), variational lower bounds (Barber and Agakov, 2003; Alemi et al., 2016), to directly approximate distributions or ratios (van den Oord et al., 2019), or via a combination of these methods.[4] These estimators have been scrutinised and analyses have shown that they offer poor theoretical guarantees and have major limitations to estimate the true value of the mutual information (McAllester and Stratos, 2020a). Nonetheless, some estimators have started to become applicable because in some tasks a comparison between the amount of information in two or more variables might be enough and it may not be required to provide an estimate of the mutual information.

For our purposes we are going to use InfoNCE (van den Oord et al., 2019) to find a lower bound on the mutual information (Poole et al., 2019a) for the second term of expression 6.9, which then we can learn through a neural network.

6.4.1 Noise Contrastive Estimation for World Modelling

InfoNCE was originally conceived for representation learning in sequences such as audio, text, or images (van den Oord et al., 2019) and it is based on Noise

[4] For a detailed overview of these estimators we refer the reader to Poole et al. (2019a).

Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2010). We refer the reader to appendix H for a general overview of the method. The basic idea behind NCE consists in given x_i from $p(x)$ and target samples y_i from a distribution $p(y|x)$ we form a *positive* pair (x_i, y_i). Then from a different *noise* distribution $p(y)$ we sample y_j to get a *negative* pair (x_i, y_j) where $i \neq j$. The objective is then to train a neural network that assigns large values to positive pairs and low values to negative pairs with a score function $f(x, y)$. The assumption here is that large scores indicate correlations between x_i and y_i. Thus the network acts as a classifier and has to discriminate when a pair is positive or negative by learning the differences between the samples from $p(y|x)$, and those coming from the noise distribution $p(y)$ (i.e. the marginals $p(y)p(x)$). InfoNCE takes this idea to generalise it to multiple samples and maximises the following function,

$$I_{NCE} = \mathbb{E}\left[\log \frac{f(x_i, y_i)}{\sum_{j \neq i} f(x_i, y_j)}\right] \tag{6.10}$$

for this expression, N data points $(y_i, ..., y_N)$ are randomly sampled. One of them is a positive pair (x_i, y_i) where $y_i \sim p(y|x)$ while $N - 1$ are negative pairs (x_i, y_j) from a product of marginals $p(y)p(x)$. That is, y_j are sampled from the noise distribution $y_j \sim p(y)$.

Now let us translate how we can apply this loss function to our specific scenario. We sample N chunks of observation and action trajectories $\{(a_{t-1}, o_t, ..., a_T, o_{T+1})\}_i^N$, take the sub-chunks $(a_{t-1:T-1}, o_{t:T})$ and obtain their corresponding latent state trajectories $z_{t:T}$. We then extract the next step sub-chunks of observations $(o_{t+1:T+1})$ and obtain their low dimensional embeddings $(\phi_{t+1:T+1})$ to maximise the following I_{NCE} expression,

$$I_{NCE} = \sum_{i}^{N*T} \log \frac{f([z_i, a_i], \phi_i)}{\sum_{j \neq i}^{N*T} f([z_i, a_i], \phi_j)} \leq I([Z_t, A_t]; \phi_{t+1}) \approx I([Z_t; A_t]; Z_{t+1}) \tag{6.11}$$

this objective provides us with a lower bound on the mutual information $I([Z_t, A_t]; \phi_{t+1})$ which is the practical equivalency of the second term in 6.9. In the expression above, $[z, a]$ indicates a concatenation between latent states and actions. For the positive pairs, ϕ_i corresponds to the low dimensional embedding produced by an encoder at the next time step. Meanwhile for the negative samples ϕ_j we can reuse the rest of the data points sampled from the buffer. $f()$ corresponds to the score function,

$$f([z_t, a], \phi) = \exp([z, a]^T W \phi) \propto \frac{p(\phi|[z, a])}{p(\phi)} \tag{6.12}$$

a log-bilinear model which is proportional to the density ratio and therefore approximates a lower bound to the mutual information $I([Z_t, A_t]; \phi_{t+1})$. The score function learns to identify positive and negative pairs through the weight matrix W. To maximise I_{NCE}, then the numerator must be large while the denominator remains as small as possible.

6.4.2 A Non-reconstructive State Space Model

In the previous section we found a way to approximate a lower bound to the second term in 6.9 via the expression 6.11. Approximating it will lead to learn a representation that is predictive about the future. However, it is still necessary to consider the first term in 6.9 to limit the amount of information that is extracted from a current observation and the latent memory. Similarly to the maximisation of mutual information, its minimisation presents a series of challenges. The most common way to proceed is via the variational upper bound by Barber and Agakov (2003), that in our case would correspond to $\mathbb{E}_q[D_{KL}[q(z_t|o_{\leq t}, a_{<t})||p(z_t|z_{t-1}, a_{t-1})]$. Note that this is the same complexity term that we have used for the previous SSMs and that appears in the derivation of the variational bound of the data log-likelihood. Thus the full loss function

that we need to optimise is,

$$\sum_{i}^{N*T} \log \frac{f([z_i, a_i], \phi_i)}{\sum_{j \neq i}^{N*T} f([z_i, a_i], \phi_j)} - \mathbb{E}_q[D_{KL}[q(z_t|o_{\leq t}, a_{<t})||p(z_t|z_{t-1}, a_{t-1})] \tag{6.13}$$

The resulting SSM has the following components:

- Encoder $e_\phi(s_t|h_t, o_t, a_t)$
- Deterministic transition $h_t = \text{GRU}_\theta(h_{t-1}, s_{t-1}, a_{t-1})$
- Stochastic transition $p_\theta(s_t|h_t)$
- Q-value model $p_\xi(q_t|h_t, s_t, a_t)$

Since the observations are no longer used as training signals, the SSM does not maintain the observation model. For planning, we follow the same procedure described in section 5.3. We simulate N rollouts to sample from the Q-model $q_{t:H} \sim p(q_{t:H}|h_{t:H}, s_{t:H}, a_{t:H})$ and evaluate the trajectories.

Algorithm 9: PR-SSM Agent

Input: Transition model, $p(s_t|h_{t-1}, s_{t-1}, a_{t-1})$, reward model $p(r_t|h_t, s_t, a_t)$, q-value model $p(q_t|h_t, s_t, a_t)$, encoder q , log-bilinear model f **Initialise:** Replay buffer D, parameters θ, latent z_t, h_t, s_t

for $t \dots T$ **do**
- Compute $h_t = f(h_{t-1}, s_{t-1}, a_{t-1})$;
- Construct and sample $s_t \sim q(s_t|h_t, o_t)$;
- $z_{t:T} = [h_t, s_t]$;
- Act $a_t \sim \text{RHE}(z_t, p(s_t|s_{t-1}, s_{t-1}, a_{t-1}), p(r_t|h_t, s_t, a_t), p(q_t|h_t, s_t, a_t)))$;
- Observe $o_{t+1}, r_t \sim p(s_{t+1}, r_t|s_t, a_t)$;
- Append $D \leftarrow D \cup \{s', a, r\}$;
- **if** *done* **then**
  - $env.reset()$;
- **end**
- **for** $i \dots Epochs$ **do**
  - Sample trajectories $o_{t:T+1}, a_{t-1:T}, r_{t:T+1} \sim D$;
  - Extract embedding $\phi_{t:T} = emb(o_{t:T})$;
  - Extract embedding $\phi_{t+1:T+1} = emb(o_{t+1:T+1})$;
  - Compute $h_{t:T} = f(h_{t-1:T-1}, s_{t-1:T-1}, a_{t-1:T-1})$;
  - Construct and sample $s_{t:T} \sim p(s_{t:T}|\phi_{t:T}, h_{t:T})$;
  - $z_{t:T} = [h_{t:T}, s_{t:T}]$;
  - Construct $p(r_{t:T}|s_{t:T}, h_{t:T})$;
  - Compute score functions $f([z, a]_{t:T}, \phi_{t+1:T:1})$ from equation 6.12;
  - Update perceptual model by computing loss $\mathscr{L}(\theta)$ from equation 6.13;
  - Construct $p(q_{t:T-1}|s_{t:T-1}, h_{t:T-1}, a_{t:T-1})$;
  - Construct and sample $q_{t+1:T} \sim p(q_{t+1:T}|s_{t+1:T}, h_{t+1:T})$;
  - Compute $q(z_{t:T-1}, a_{t:T-1}) = r_{t+1:T} + \gamma \arg\max_a q(z_{t+1:T}, a_{t+1:T})$;
  - Update q-value model by computing loss $\mathscr{L}(\theta)$ from equation 5.2;
- **end**

end

6.5 Experiments

6.5.1 Four-Room tasks

We consider the same four task variations of the four-room environment introduced in chapter 5. We compare the non-reconstructive SSM (NR-SSM) that we have just described against the best-performing reconstructive agent from the previous chapter. Hence, both agents evaluate a potential plan by obtaining the cumulative sum of Q-values from a simulated trajectory. We will refer to the reconstructive agent simply as SSM. In addition, we also compare these agents against Dreamer v2 (Hafner et al., 2022) and MuZero (Schrittwieser et al., 2020) as they are two representative state-of-the-art architectures in model-based RL research. Due to the large resources required by those two agents, we report the results obtained from training them during a period of 12 hours (by comparison, our agents take from one to four hours to complete their training for 1000 episodes using the same computational resources). We have taken publicly available PyTorch implementations and have adapted them to work with our environments[5,6,7].

In level 1, except for a minor period of instability, the SSM and NR-SSM agents are comparable in their average return over time (fig. 6.1 left). Let us remember that the first level only terminates either due to a time limit or when the agent reaches the goal state. Thus we can also analyse the performance of these agents from the point of view of the effort in terms of the number of time steps required to optimise their behaviour. Right fig 6.1 depicts the cumulative number of steps across episodes. It can be observed that there are abrupt changes in the time step growth rate for both agents. Although they learn the task in a short number

[5] Where necessary we have also modified the hyperparameters to follow those reported in the original publications.
[6] Dreamer v2: https://github.com/RajGhugare19/dreamerv2
[7] MuZero: https://github.com/werner-duvaud/muzero-general and https://github.com/Hauf3n/MuZero-PyTorch

of episodes, during the early training stages the agents have not learned yet the most direct paths that lead to the goal. Once this occurs, the rate of change of the total number of time steps decreases. The NR-SSM agent exhibits a higher level of efficiency. For the SSM agent, it takes almost double the number of steps for this transition to occur.

We also observed similar performance between these two agents in level 2 (fig. 6.2). Although the NR-SSM agent achieved mastery over the task slightly faster than the SSM agent. In the more challenging level 3, it is where we can notice more differences in their performances. We noted in the previous chapter that the SSM maintains a gradually increasing trend. In the case of the NR-SSM, its improvement occurs more suddenly. The differences in improvement can also be analysed from the perspective of the success rate (fig. 6.3 center left) or the cumulative return (fig. 6.3 bottom left). The average overall performance indicates that the NR-SSM agent reaches the goal 60% of the time compared to 50% by the SSM. We can also observe from the cumulative return plot that the NR-SSM undergoes a behavioural transition before it reaches 300 episodes. At that point, the agent starts to consistently reach the goal. The differences between the agents become more pronounced in level 4, where the SSM agent is not able to match the episodic return of NR-SSM even in the long term (fig. 6.3

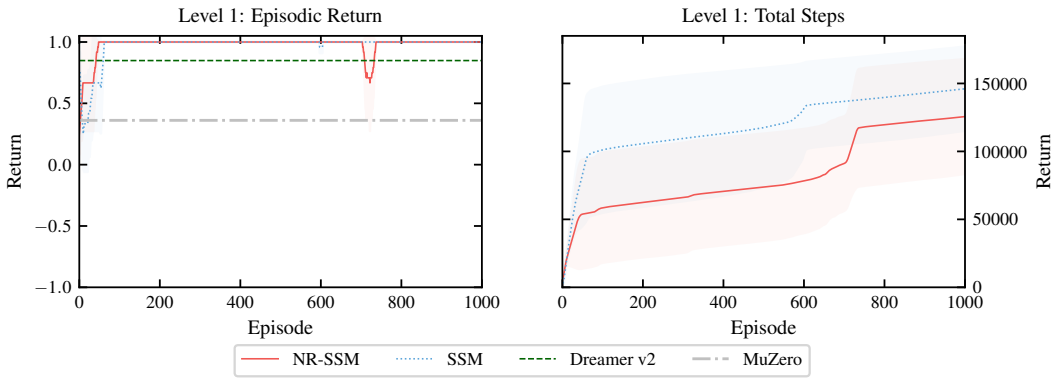


Figure 6.1: *Level 1 results. Left: episodic return. Right: cumulative number of steps in time.*

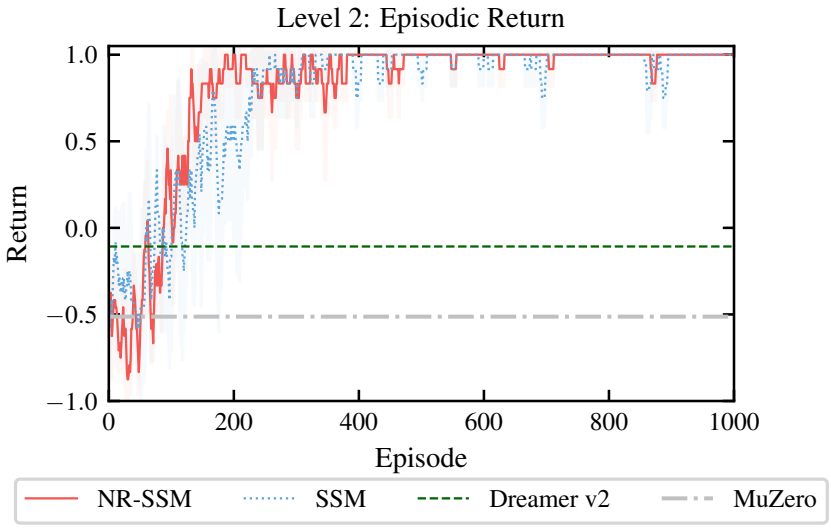


Figure 6.2: *Level 2 results. Episodic return.*

top right). We can notice from the trend in the cumulative return (fig. 6.3 center right), that as it learns, the SSM manages to slow down the trend but it does not reverse it. The NR-SSM on the other hand reverses it within 500 episodes. From thereafter, although in some episodes the NR-SSM does not reach the goal, the rising trend indicates that as training continues it can sustain a positive performance and achieve a positive reward in the majority of the episodes.

We can also observe from the results that Dreamer v2 and MuZero achieve a degree of competence in levels 1 and 2. However, it remains below those exhibited by NR-SSM and SSM. Moreover, the performance of Dreamer v2 and MuZero plummets in levels 3 and 4 where it becomes more challenging for an agent to attain a positive reward. The neural network architectures of these two baselines are considerably larger than the ones we are using, and so in principle, they can cope with environments with more complex dynamics than the smaller architectures. Nonetheless, the size of their architectures in combination with the hyperparameter choice may hurt their immediate performance in these tasks. For instance, during Dreamer v2's training, we often observed that the world model loss behaved unsteadily or increased to become unmanageable. We also tested

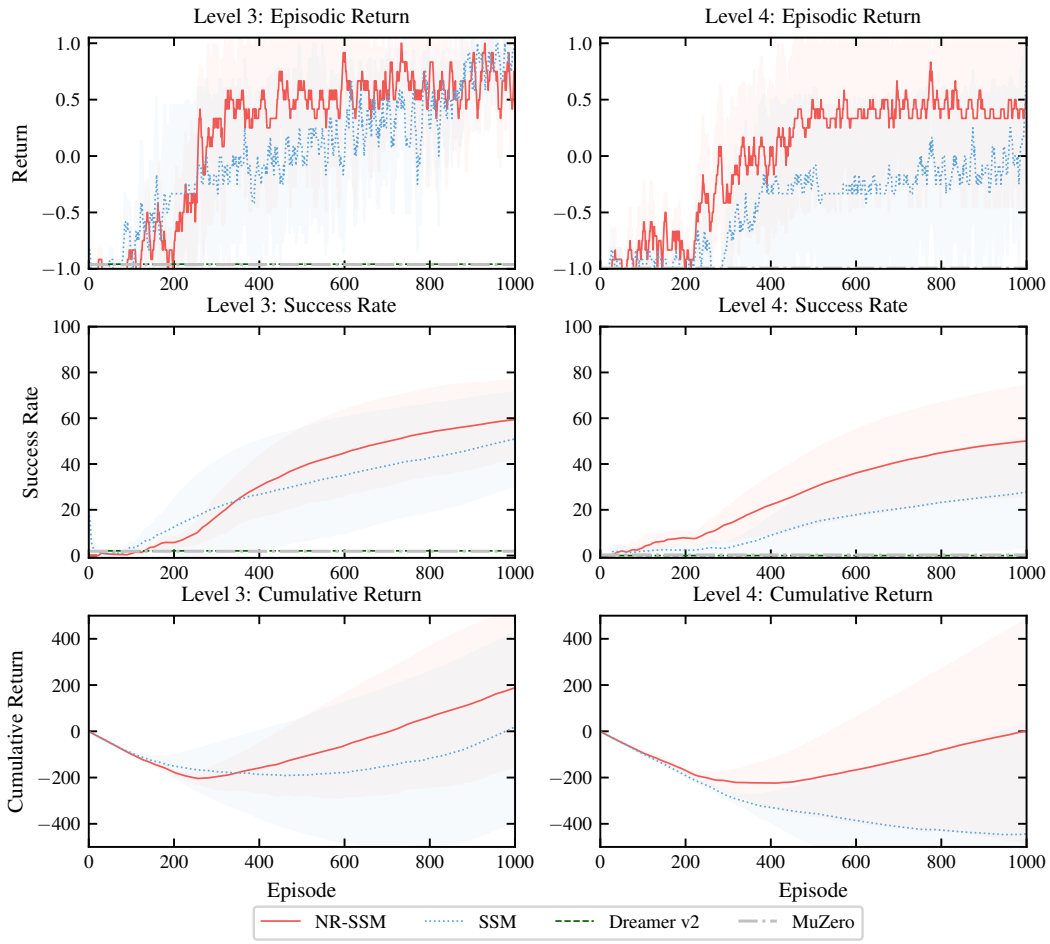


Figure 6.3: *Level 3 and 4 results. Top row: episodic return. Middle row: success rate in time. It indicates the rate of episodes in which the agent has reached the goal. Bottom row: cumulative return in time. Initially, there is a steep decline as the agent starts to learn a forward model and is unable to solve the task in most of the episodes. The trend either stabilises or inverts as training continues and the agent becomes more proficient.*

these baselines with modified neural network architectures that were comparable in size to our agents however, they still had inferior performances. In the case of Dreamer v2, some of the reasons for its performance could be because it has to learn the actor-critic over simulated experience, which may require more training steps to start to produce more accurate assessments. Meanwhile, in our agents, although we produce simulated sequences of Q-values, the Q-model is learned with latent trajectories that had been experienced by the agent. Furthermore, in Dreamer v2 this implies learning two components, the actor and the critic,

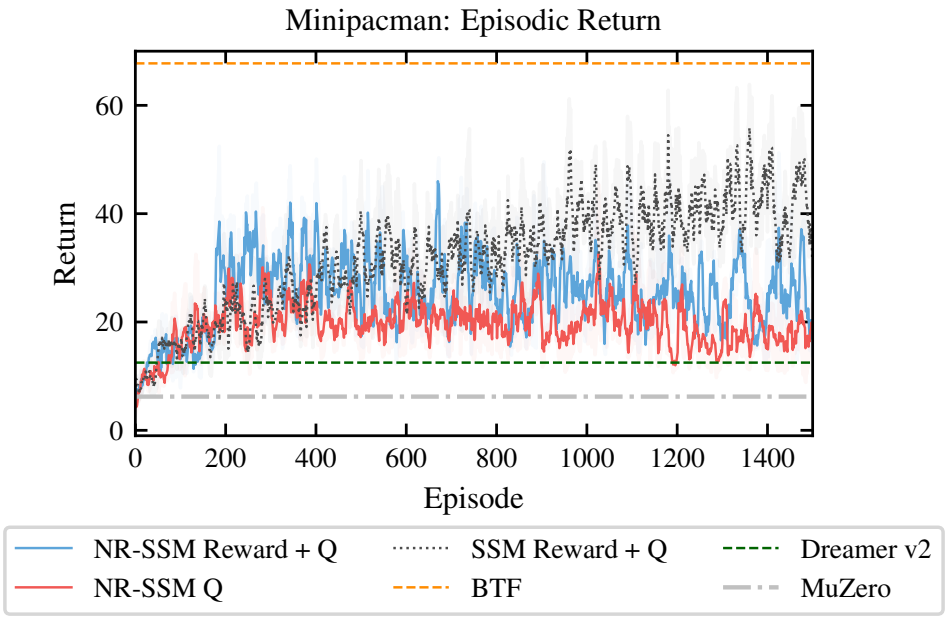


Figure 6.4: *Results in Minipacman. Episodic return.*

whereas in our agents the planner directs the process of action selection invoking the Q-model but the planner does not have to be learned. In the case of MuZero, the original implementation uses a ResNet which is considerably larger compared to the neural networks used in Dreamer v2 or this work. If we substitute the ResNet for a smaller architecture, it is possible to speculate that MuZero has more difficulties to compensate for the lack of additional rich supervision signals provided either by the reconstruction or the predictive relevance objectives. Another aspect that can be contributing negatively to its performance is that MuZero models deterministic transitions which might be insufficient to deal with the adversarial and uncertain elements in the environment.

6.5.2 Minipacman

For Minipacman we test the two variations of NR-SSM. The first evaluates the sum of rewards and the Q-value for the last step. The second variation sums the Q-values along the trajectory. In contrast to the previous task, we compare them

Figure 6.5: *Left: the frame shows the ground truth. Right: a reconstruction from a decoder trained on top of a frozen non reconstructive PR-SSM.*

against the SSM variation that gets the sum of rewards and Q-value in the last step. We also compare them against Bootstrapped Transition Functions (BTF), MuZero, and Dreamer v2. Similar to what we observed in chapter 5, the variation which selects the plans according to the sum of rewards and the last step Q-value exhibits better performance. We can also observe that the SSM obtains a higher average score in the long term. The NR-SSM Reward + Q initially starts to learn faster than the SSM but after 200 episodes it starts to fluctuate within the same range. As this might be due to the choice of training hyperparameters, and often due to the learning rates, we reevaluated with a different set of them albeit trying to keep them close to the other comparisons for fairness. We again observed the same behaviour described in chapter 5 in which the agent is confined to an area of the maze where the immediate surroundings consist of cells that have already been eaten (see figure 6.5 for an example of this situation). We have suggested that to mitigate this issue we could encourage some form of exploration in the agent. In addition, we could assess alternative representation objectives in the I_{NCE} term of the NR-SSM that could emphasise local elements of the environment (Anand et al., 2020). That is instead of approximating a lower bound to the mutual information between global features to conduct it between local features or global to local features.

6.6 Discussion

In this chapter we revisited the idea of learning forward models following reconstructive learning objectives. Based on a series of converging threads, we argued for the potential of exploring training objectives beyond the reconstruction of external scenes. We proposed an objective grounded on three main assumptions. The first is that most of the data stream that passes through the agent's sensors does not contribute to the prediction of the future. From this first claim we then argue that an agent should concentrate on the prediction of a subset of the future rather than on whole observations. The third assumption, is that a latent state that is smaller than an observation captures some of its features and can be a candidate for the prediction target.

From these assumptions then our goal was to construct a learning objective that could encourage an agent to actively discard information from the past by prioritising the retention of information that is considered to be substantial for the prediction of the future. Importantly, here the future that concerns the agent is defined to be the embeddings in latent state space. To formulate the training objective we take as a basis two concepts from the information-theoretic literature: the information bottleneck and predictive information.

Let us examine more closely the relation between our objective and these two concepts. The basic premise behind predictive information is to quantify how correlated is the past and future of a process. It hints that most of the observations are non-consequential for the prediction of the future. If we adhere strictly to its mathematical definition, the past and future data stream are in the same space. Thus it has been proposed as a measure of the complexity of a process (Shalizi and Crutchfield, 2001). In our case, we are not interested in the complexity of the agent-environment coupling or the complexity of the observation process. Instead, our interest is in the internal perspective of an agent that is trying to acquire a forward model that can sustain its behaviour. Thus the translation of

these ideas to our scenario comes with some nuances. If most of X_{past} is irrelevant for predicting X_{fut} , the same will be true for X_{fut}. Once we observe a part of X_{fut} (and thus integrated within X_{past}) most of it has to be discarded to predict the part of X_{fut} that remains still unseen in the future. Therefore an agent that intends to be resource efficient should not aim to predict X_{fut} but only a subset of it. This takes us to the second point on extracting information that is relevant to predict a subset of X_{fut}. The notion of relevance that we have invoked presupposes that a target of interest (i.e. a relevant variable) has been defined. If that is the case, then during the optimisation process it is possible to extract the information that is relevant to predict this target. It can be noticed then that there is a more fundamental outer problem of relevance that concerns the definition of the target of interest itself. That is, to identify the information that is relevant for a target it is first necessary to determine a *relevant target* variable. Whether or how this process of discovery could be autonomously achieved is at this moment a problem we have left unexplored. Instead, as we have described, the agent's future latent state has been predefined as the relevant target. In chapter 8.1.4 we will speculate about other targets that we could consider.

Regarding the results we obtained. In the four-room tasks we observed not only better performance when using the non-reconstructive objective but also, in those tasks where we can measure the steps it takes to reach a given performance under equal conditions, we noted a higher level of efficiency. In Minipacman, however, we did not observe this increase in performance. To analyse the representation learned by the architecture, we froze it and trained a decoder on top of the learned latent state. We noticed that the model still captures a large part of the observation. There are different speculations we can make from the analyses. For example, the architecture's latent space might be too large which leads to excess capacity to store information. Another issue could be associated to the mutual information approximation with the bounds not being tight enough (McAllester and Stratos, 2020b; Poole et al., 2019b; Song and Ermon, 2020). A

third speculation could be due to the network features we included as part of the mutual information maximisation. As mentioned in the previous section, an alternative could be to maximise the information between global and local features. Further experiments in different benchmarks could also help to elucidate the type of environments in which this learning objective results advantageous.

7 Learning and planning from self-regulating signals

7.1 Introduction

In the previous chapters the role of the model during planning has been to provide next state predictions to generate the rollouts, and rewards or Q-values to evaluate them. Although it is certainly true that because the models generate predictions, they have a central role in guiding the behaviour of the agent, we have not explored, for instance, the properties of those learned models or the relations encoded by them. Intuitively, the model reflects the experience accumulated by the agent and its particular coupling with the environment. Two different agents in the same environment, may learn different forward models due to their specific modes of engagement. Even in the same agent, the predictions generated by its model might be different when presented with the same observation if the model is at different epistemic states. For example, when the model is in the early, and then, in late stages of training. This subjective quality of the interaction of an agent with its medium is also captured in debates regarding the interpretation of reward as a construction of the agent dependent on its motivational states or knowledge (Dayan, 2012; Juechems and Summerfield, 2019). Several lines of research in the perception-action loop, which we briefly review in the next section, attempt to ground some form of utility or reward within the agent. Our objective here is to try to connect this theme to the larger story of approximating an internal model through the framework of *active inferece* (Friston et al., 2012a).

This framework is closely related to model-based Reinforcement Learning (RL). However, it is conceived to establish the whole interaction between the agent and the environment as a *belief-based* interaction. This perspective allows us not only to use the model as a tool to generate predictions, but also to understand it as a reference point from which to provide a contextual assessment of the value of an observation. To be more concrete, any incoming observation is contrasted against a specific distribution. Active inference tells us that this distribution is a prior describing the preferences of an agent. If the observation is deemed to be probable under this distribution then it is considered to have a low information-theoretical *surprisal*. Active inference assumes that an agent acts in order to seek observations with low surprisal. This view is controversial (Herreros and Verschure, 2015; James, 2015; Paolo et al., 2022; Aguilera et al., 2022) however a full discussion on the subject is beyond the scope of the thesis. The main point to take for the moment is that the statistical relationships captured by a model, (1) ground the value of an outcome within the agent and (2) serve as an anchor to actively guide the behaviour of the agent.

Our concern here in this chapter is the emergence of the distribution of preferences. Since a major theme in the thesis is the construction of internal models, and the distribution of preferences is a subset of the larger internal model, the crucial question is what it should correspond to? The active inference literature does not elaborate on its origin or how it arises. Moreover, it shows inconsistency regarding the outcomes that have to exhibit low surprisal, with some works considering observations (Friston et al., 2016), hidden states (Friston et al., 2009) or internal states (Friston, 2012).

We propose a scheme, where the main assumption is that an agent has access to external and internal inputs. We define the internal inputs as those that have to exhibit low surprisal and derive a new mathematical expression. We then show an example of how the preference distribution arises in a self-supervised manner and how the agent's behaviour reinforces it. For planning, we integrate Rolling

Horizon Evolution (RHE) as part of the active inference agent. The evaluation of the rollouts is given by the predicted surprisal and predicted information gain, that is, the *expected free energy*.

To contextualise our contribution, we start by providing a review of active inference and the particular issues surrounding the definition of the preference distribution. For completeness we also show how we can transition from RL to the belief-based account of active inference, and describe a model-free surprisal minimisation agent originally proposed by Berseth et al. (2020) which serves as our baseline.

7.2 Preferences, desires and surprisal minimisation

Active inference (Friston et al., 2012a) provides an alternative formulation to the problem of sequential decision-making in Markov Decision Process (MDP). Its origins can be traced to studies connecting machine learning ideas to statistical physics (Hinton and Zemel, 1993; Hinton and van Camp, 1993; Dayan et al., 1995) and in variational inference (Jordan et al., 1999; Ghahramani and Beal, 2000; Beal, 2003). However, the framework of active inference itself has largely been developed in theoretical neuroscience as a probabilistic account to describe perceptual mechanisms in the brain (Friston et al., 2006) and goal-seeking behaviour in living processes (Friston, 2012).

Active inference shares many similarities with model-based RL and in particular with proposals that have formulated control as an inference problem (Attias, 2003; Todorov, 2008; Rawlik et al., 2010; Botvinick and Toussaint, 2012; Kappen et al., 2012). One of the major differences between active inference and RL is the conceptualisation of the utility. RL borrows from animal learning theories in psychology the notions of trial-and-error learning and *secondary reinforcers*.

These are the stimuli that are associated with primary reinforcers such as food, water, or pain. Thus, in RL the utility is interpreted as a reward signal that provides positive or negative reinforcements and guides the agent's behaviour toward a goal. As we know, the objective in RL is to maximise the expected sum of rewards. This is known as the *reward hypothesis* (Sutton and Barto, 2018) (cf. dynamic programming methods in optimal control in which the utility is interpreted as a cost that a controller has to minimise, for example, energy expenditure). The standard formalism of RL considers the reward as a signal received *by* the agent *from* the environment. Accordingly, the notion of reward is ingrained as something that occurs externally and independently from the agent. This is emphasised by the fact that historically RL has been oriented towards achieving competence in a task, and rewards tend to be treated as part of the problem specification defined by the researcher. The problem of reward sparsity, for instance, arises in part because of the difficulty of defining scalar rewards for complex, hierarchical or ambiguous behaviour. In this sense, standard RL leaves unaddressed central concerns about the nature of behaviour.

Nonetheless, let us note that there several lines of research in the perception-action loop trying to characterise generic and task-independent drives as grounded within the agent. In a body of work known as *intrinsic motivation*[1] which has been defined as an activity performed for its inherent satisfaction (Ryan and Deci, 2000), we can find computational models that relate curious exploratory behaviour to the maximisation of learning progress (Oudeyer and Kaplan, 2007; Oudeyer et al., 2007). Closely related to it but from an information-theoretic perspective, is the maximisation of predictive information[2] between consecutive observations which gives rise to behaviours that generate dynamics that are rich in information content while remaining predictable (i.e learnable) (Ay et al., 2008).

[1] See Biehl et al. (2018) for a perspective on intrinsic motivation under the active inference framework.

[2] Predictive information (Bialek and Tishby, 1999) is an important influence in the construction of the architecture presented in chapter 6. Our treatment, however, is not from the perspective of a feedback signal but rather as a learning objective for the construction of a world model.

Other information-seeking behaviours have been characterised as information gain measures (Little and Sommer, 2011; Mobin et al., 2014). Meanwhile, other agent-centric approaches have formalised insights from biology and ecological psychology. For instance, *empowerment* measures the control or influence in the world as perceived by the agent (Klyubin et al., 2005; Salge et al., 2014). Some of these principles and proposals have been imported into RL under the term of *intrinsic rewards*. Some examples include, rewards derived from prediction error (Jaderberg et al., 2016; Pathak et al., 2017), novelty (Bellemare et al., 2016; Burda et al., 2018; Ostrovski et al., 2017; Henaff et al., 2022), information gain (Houthooft et al., 2016), empowerment (Mohamed and Rezende, 2015; Kumar, 2018) or ensemble disagreement (Pathak et al., 2019). It is important to highlight that in RL these intrinsic rewards are treated as surrogate or complementary rewards in the absence of dense external reward signals. They are often intended to encourage exploration to increase the prospects of gaining competence in a particular task.

In active inference, instead of maximising rewards, the agent minimises a quantity called the variational *free energy* which is explained in detail in this chapter. The agent tries to reduce the difference between sensations and predictions. The agent holds a prior over preferred future outcomes, thus the agent evaluates to what extent the outcomes it observes align or not with those preferences. Therefore value arises not as an external property of the environment, but instead, it is contextually conferred by the agent depending on its current configuration and the interpretation of stimuli.

From an algorithmic perspective, this idealised account of active inference presents some challenges. Active inference does not elaborate on the origin of the prior preferences or their characteristics. Because the framework is motivated by biological considerations, these priors are assumed to emerge over evolutionary scales or during a lifetime.

Nonetheless, active inference has recently garnered interest in machine learning, as researchers have adapted the framework to solve control and decision-making tasks. Similar to the external rewards in RL, the prior distribution becomes part of the problem specification. Thus the objectives of the task are also predefined beforehand. This suggests that much of the effort that can go into reward engineering in RL is reallocated to the construction of a distribution that specifies preferred outcomes. In this chapter, we explore very basic steps to move away from this requirement and to allow the agent to dynamically discover this unknown distribution.

The approach we follow is inspired by the literature on the foundations of adaptive behaviour and organismic autonomy (Di Paolo, 2005, 2010; Barandiaran et al., 2009). We consider the emergence of primitive forms of self-supervised behaviour as a response to the perturbations an agent must endure to maintain its structural integrity. More concretely, we examine how can a signal acquire functional significance as the agent identifies it as a condition necessary for its viability and future continuity in the environment. We will see that, because in active inference we can formulate the agent-environment interaction under the same probabilistic language, we can treat all utilities as beliefs grounded on the agent rather than external and detached quantities. To ease the transition from a reward-maximisation to a surprisal-minimisation view of behaviour, first, we start by introducing the surprise minimising RL (SMiRL) specification (Berseth et al., 2020) before we proceed with a brief overview of the expected free energy. Then we motivate our approach from the perspective of a self-regulating organism. Finally, we present the results from our case study.

7.3 Model-free surprisal minimisation

To illustrate the concept of belief-based utility and start introducing our approach, we begin by describing a generic outline of a model-free Surprise Minimizing

Reinforcement Learning (SMiRL) (Berseth et al., 2020) agent. Consider an environment whose generative process produces a state $s_t \in \mathcal{S}$ at each time step t resulting in an agent observing $o_t \in \mathcal{O}$. Note the slight difference concerning the classic RL formulation reviewed in section A.1. For a belief-based agent, o and s are distinguished from each other because o corresponds to the observations gathered by the agent and these may not necessarily be the same as the hidden causes s. An example of this occurs also in Partially Observable Markov Decision Process (POMDP) RL formulations, where the agent holds a belief b about the state of the world s. The agent acts on the environment with $a_t \in \mathcal{A}$ according to a policy π, and obtains the next observation o_{t+1}.

Thus far, the general picture resembles standard RL formulations. Now suppose that the agent performs density estimation on the last $t - k$ observations it has gathered, and from this process, it obtains a current set of parameter(s) θ_t summarising the density $p_\theta(o)$. That is, the agent now also holds beliefs about the type of observations it has gathered in the past. Because these sufficient statistics θ contain information about the agent-environment coupling, they are concatenated with the observations gathered from the environment to form an augmented state $x_t = (o_t, \theta_t)$. At every time step, when the agent receives a new observation, it then computes the surprisal produced by this new observation given the current estimate encoded in θ. The agent also performs a new iteration of density estimation and updates θ. Unlike in standard RL, instead of maximising a reward, the agent maximises the log of the model evidence

$$\max_{\pi} \mathbb{E}_{\pi}\left[\sum_{k}^{K} \gamma^k \log p_{\theta_{t+k}} | s = s_t, a = a_t\right] \tag{7.1}$$

maximising the log evidence is equivalent to minimising surprisal. Therefore, we decide to maintain consistency with active inference and rewrite the Bellman

equation for a value-based method such as Q-learning in the following manner

$$Q_{\pi^*}(x, a) = \mathbb{E}_{\pi}[-\log p_{\theta}(o) + \gamma \min_{a'} Q_{\pi^*}(x', a')] \tag{7.2}$$

this expression shows that the agent optimal policy will tend to favour actions that lead to a reduction in surprisal. The Q-function can be estimated as normal via Deep Q-Network (DQN) (Mnih et al., 2013) or any other function approximator parameterised by ϕ such that $Q_{\pi^*}(x, a) \approx Q(x, a; \phi)$.

7.4 Active Inference

The *Free Energy Principle (FEP)* (Friston et al., 2006) has evolved from an account of message passing in the brain to propose a probabilistic interpretation of self-organising phenomena (Friston, 2013; Ramstead et al., 2019, 2018). Given the broad scope of the theory, our discussion is centred on active inference where it is possible to motivate an account of the perception-action loop at the same level as RL.

The FEP posits the view that a system remains far from equilibrium (i.e. alive) by maintaining a low entropy distribution over the states it occupies during its lifetime. Accordingly, the system attempts to minimise the surprisal of an event at a particular point in time. To understand this more concretely, recall the density $p(o)$ from the previous section. In active inference, it is assumed that $p(o)$ encodes the states, drives, or desires the system should fulfil. Thus the system strives to obtain or observe outcomes o that minimise the surprisal $-\log p(o)$. This, also as discussed in the previous section, is equivalent to maximising the model evidence or marginal likelihood $p(o)$.

The estimation of the actual marginal is intractable. Active inference confers this technical fact a naturalistic interpretation based on perceptual grounds. Namely,

an agent cannot estimate $p(o)$ exactly due to the inability to have complete knowledge of the structure and causal dynamics of an environment. Therefore an agent instead resorts to minimising a quantity called *free energy* (Hinton and Zemel, 1993; Dayan et al., 1995). Mathematically, the free energy is no other than an upper bound on surprisal (Jordan et al., 1999) and we can derive it using the principles of variational inference reviewed in A.3.

$$-\log p(o) = -\log \mathbb{E}_{q(s)}\left[\frac{p(o,s)}{q(s)}\right] \leq \underbrace{-\mathbb{E}_{q(s)}\left[\log \frac{p(o,s)}{q(s)}\right]}_{Free\ Energy} \tag{7.3}$$

where $p(o,s)$ is the generative model (i.e. forward/predictive/world model) of the agent and $q(s)$ the beliefs encoded in the variational density approximating hidden causes. The right-hand side corresponds to the free energy which is the same as the negative Evidence Lower Bound (ELBO) (equations in A.20). Rearranging 7.3 we obtain

$$\mathrm{F} = \mathbb{E}_{q(s)}[\log q(s) - \log p(o,s)] \tag{7.4}$$

because the free energy is a $D_{KL}[q(s)||p(o,s)]$ then an agent minimising the free energy is also approximating the true posterior of hidden causes through a variational density expressed as $D_{KL}[q(s)||p(s|o)]$ (equations and inequalities in A.23 and A.22). The name of free energy is given due to the similarities to the Helmholtz free energy in thermodynamics, which can be appreciated more clearly by rearranging equation 7.4

$$\begin{aligned} \mathrm{F} &= \underbrace{-\mathbb{E}_{q(s)}[\log p(o,s)]}_{Energy} + \underbrace{\mathbb{E}_{q(s)}[\log q(s)]}_{(Negative)\ entropy} \\ &= U - TS \end{aligned} \tag{7.5}$$

where U corresponds to the internal energy of the system, S to the entropy and T to the temperature of the surroundings (here $T = 1$).

7.4.1 Expected Free Energy

Equation 7.4 is used to compute a static form of free energy and infer hidden causes given a set of observations. However, if we instead consider an agent that acts over an extended temporal dimension, it must infer and select policies that minimise an Expected Free Energy (EFE) (Friston et al., 2016) of a policy π for a future step $\tau > t$. Conditioning the free energy over policies we can express the expected free energy G as

$$\mathrm{G}(\pi, \tau) = \mathbb{E}_{q(o_\tau, s_\tau|\pi)}[\ln q(s_\tau|\pi) - \ln p(o_\tau, s_\tau|\pi)] \tag{7.6}$$

where $p(o_\tau, s_\tau|\pi) = q(s_\tau|o_\tau, \pi)p(o_\tau)$ is the generative model of the future. Rearranging G as,

$$\mathrm{G}(\pi, \tau) = -\underbrace{\mathbb{E}_{q(o_\tau|\pi)}[\ln p(o_\tau)]}_{Instrumental\ value} - \underbrace{\mathbb{E}_{q(o_\tau|\pi)}\left[D_{KL}[\ln q(s_\tau|o_\tau, \pi)||\ln q(s_\tau|\pi)]\right]}_{Epistemic\ value} \tag{7.7}$$

illustrates how the EFE entails a first term known in the active inference literature as *pragmatic, instrumental* or *goal-seeking*. It provides value to the agent to the extent that it can fulfil its preferences encoded in $p(o)$. Then there is a second term referred to as *epistemic* or *information seeking* which provides value by resolving uncertainty. To make an analogy with RL, instrumental value encourages exploitative behaviour (i.e. externally defined rewards) and epistemic value encourages exploratory behaviour (i.e. intrinsic rewards).

In summary, an agent which minimises its free energy via active inference can either change its beliefs about the world or sample the regions of the space that conforms to its beliefs, desires, or goals.

7.5 Self-regulating adaptive control

We have discussed how $p(o)$ is a crucial piece in the active inference formalism. In which one of its interpretations corresponds to a distribution that encodes the goals, needs, or objectives of the agent. In contrast to a reward, it emerges within a context that is more mathematically principled. However, there is a gap between the appealing interpretation and the algorithmic specification. From the point of view of an organism, the framework argues that if an agent acts to minimise its surprisal it will orientate its behaviour towards a set of states in which the agent or the organism is viable. But an aspect that is less understood is how these attracting viable states come into existence. That is, how do they emerge and are selected from the particular conditions surrounding the system, and how are they discovered among the potential space of signals and the myriad of internal and external information channels? The studies that have attempted to provide algorithmic implementations of active inference have faced the challenge of how to specify $p(o)$. Their approaches have consisted in using the reward given by the environment (Friston et al., 2012a; Ueltzhöffer, 2018; Tschantz et al., 2019, 2020; Sajid et al., 2020), encoding it in $p(o)$ as $p(o) \propto \exp(r)$ (Millidge,

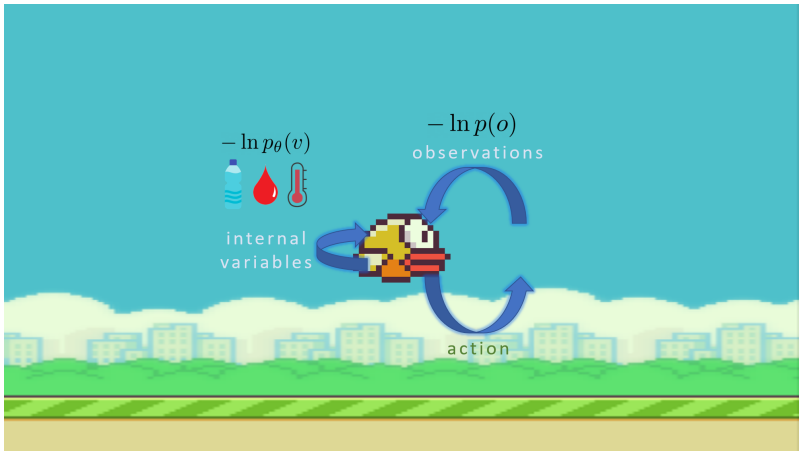


Figure 7.1: *An extended idealised version of the perception-action loop. In addition to the classic observation and action channels, the agent also gathers other readings informing it about its internal state. An example of this in biological organisms is the levels of oxygen, blood pressure, or temperature. Note that we do not define these viability variables and instead propose a simpler set-up with a single binary variable indicating the structural integrity of the agent.*

2020) or passed on through a hierarchical predictive coding scheme (Millidge, 2019). While others have relied on demonstrations (Çatal et al., 2019, 2020; Sancaktar et al., 2020). Therefore, similarly to RL, it has been assumed that there is pre-existent knowledge of the goals and the rewards an agent must attain.

For the work presented in this chapter, we take inspiration from the homeostatic and regulatory properties of living processes, as a step towards the study of more generic and universal signals from which complex behaviour may emerge. The main idea behind our approach is to differentiate between the nature of the signals that an agent has access to, such as the sensory signals induced by external stimuli, and the internal signals inherent to the agent and caused by interoceptive states. For example, figure 7.1 shows an idealised version of the perception-action loop. With a channel dedicated to the actions and another for receiving external stimuli o. In addition, the agent has an internal channel from which it obtains information related to its internal configuration. We denote these internal readings as v. Instead of fixing a predefined distribution of preferences $p(o)$ (or $p(v)$ in our case) as it has been done in the active inference literature,

the preferences guiding the behaviour of the agent are learned (and can change) dynamically by performing density estimation to obtain parameters θ. In our scenario, we make the following assumption. If the agent is equipped only with basic density estimation capabilities (e.g. low capacity for representation) or the environment is complex and has fast-changing dynamics, then it becomes difficult for the agent to structure its behaviour around the type of regularities in observation space that can sustain it in time. In these situations, rather than minimising free energy over sensory signals o, the agent may instead leverage these sensory signals to maintain low surprisal on another target variable (e.g. v). This implies that although the agent may have in principle access to multiple signals, it might be interested in maintaining only some of them within a certain expected range and low entropy.

In a biological organism, an internal channel may contain information about the essential variables of the system such as oxygen levels, blood pressure, or temperature. For simulated agents, defining what should constitute the artificial physiology or the internal states is hardly understood or researched. Therefore we assume the introduction of a communication channel from which the agent retrieves a signal that informs it about its continuity in the environment. For the moment, we have defined this signal in abstract terms as simply a binary signal indicating the presence of the agent in the task. We can make a rudimentary comparison, and think of this quantity in a similar way to how feelings such as hunger or pain agglutinate and coarse-grain the changes of several internal physiological responses (Damasio, 2004). The approach is generic, in that it is agnostic to the content or the number of signals it could consider. Moreover, the magnitude of the numerical value of the signals is irrelevant by itself and it only acquires significance for the agent as it becomes statistically associated with its viability.

7.5.1 Expected Free Energy via Rolling Horizon

We introduce a novel variation of RHE for generating action-sequences and associating them to an EFE score. Similarly to the RHE presented in section 3.5, it interoperates with learned approximate forward models and can handle ensembles. However, we also made some technical improvements to detach the observation transitions from the EFE transitions. This allowed us to evaluate more action-sequences and expedite the operation, as they were processed in parallel in a single call.

The RHE objective becomes

$$\pi(s_0) = \min_{\pi=a_{\tau:T}} \sum_{\tau>t}^{T} \mathrm{G}(\pi, \tau) \tag{7.8}$$

that is, RHE searches in the space of policies π for the one that minimises the expected free energy G along a trajectory. Alternatively, we can specify a probabilistic formulation of RHE consistent with the active inference framework

$$q(\pi(s_0)) = \sigma\Big(- \beta \sum_{\tau>t}^{T} \mathrm{G}(\pi, \tau)\Big) \tag{7.9}$$

where σ is the softmax function. In this case, action-sequences are selected in proportion to their G and the magnitude of the inverse temperature β controlling the stochasticity. Note that we define a new functional for G that considers the agent's internal channel (appendix C)

$$\mathrm{G}(\pi, \tau) \approx -\mathbb{E}_{q(o_\tau, v_\tau, \theta|\pi)} D_{KL}[q(s_\tau|, o_\tau, v_\tau, \pi)||q(s_\tau|\pi)] - \mathbb{E}_{q(v_\tau, \theta, s_\tau|\pi)}[\ln p_\theta(v_\tau)] \tag{7.10}$$

Moreover, if we explicitly consider the model parameters ϕ, equation 7.10 can be decomposed as (appendix D),

$$
\begin{aligned}
\mathrm{G}(\pi, \tau) \approx & -\underbrace{\mathbb{E}_{q(o_\tau, v_\tau, \phi|\pi)} D_{KL}[q(s_\tau|o_\tau, v_\tau, \pi)||q(s_\tau|\pi)]}_{salience} \\
& -\underbrace{\mathbb{E}_{q(o_\tau, v_\tau, s_\tau|\pi)} D_{KL}[q(\phi|s_\tau, o_\tau, v_\tau, \pi)||q(\phi)]}_{novelty} \\
& -\underbrace{\mathbb{E}_{q(o_\tau, v_\tau, s_\tau, \phi|\pi)}[\ln p_\theta(v_\tau)]}_{instrumental\ value}
\end{aligned}
\tag{7.11}
$$

The expression unpacks further the epistemic contributions to the EFE in terms of salience and novelty (Friston et al., 2017). These terms refer to the expected reduction in uncertainty about hidden causes and the parameters respectively.

Algorithm 10: Active inference agent with internal states

Input: Observation model $p(o_{t+1}|o_t, a_t)$, internal measurement model $p(\theta_{t+1}|o_t, a_t)$

Initialise: Replay buffer D, interoceptive buffer B, observation o_t, internal measurement v_t, parameters θ_t, ϕ

for $t \dots T$ **do**
  Act $a_t \sim \mathrm{RHE}(o_t, \mu_t)$;
  Observe $o_{t+1}, v_{t+1} = env.step(a_t)$;
  Fit density $\theta_{t+1} = mean(B)$;
  Compute surprisal $r_t = -\log p(m_{t+1}; \mu_{t+1})$;
  Append $B \leftarrow B \cup \{m_{t+1}\}$;
  Append $D \leftarrow D \cup \{o, \theta, a, r, o', \theta'\}$;
  if *done* **then**
    $env.reset()$;
    Sample $o, \theta, a, r, o', \theta' \sim D$;
    Sample $o' \sim p(o'|o, a)$;
    Sample $\theta' \sim p(\theta'|o, a)$;
    Update model parameters ϕ by computing negative log likelihood;
  end
end

Algorithm 11: Rolling Horizon Evolution planner with Expected Free Energy evaluation

Input: Forward model p_ϕ, number of actions A, sequence length $\mathcal{T}$, population size $\mathcal{P}$, mutation rate η

if *not shift buffer* **then**
  $\pi \leftarrow a_t, \ldots, a_{\mathcal{T}} \sim Cat(A)$;
else
  Shift π to the left and add $a_{\mathcal{T}} \sim Cat(A)$ at the end of the sequence;
end
$\pi \leftarrow$ Mutate π with rate η;
for $t \ldots \mathcal{T}$ **do**
  Simulate to construct $p(o'|o, \theta, a)$ and $p(\theta'|o, \theta, a)$;
  Sample $o' \sim p(o'|o, \theta, a)$;
  Sample $\theta' \sim p(\theta'|o, \theta, a)$;
  Compute Expected Free Energy G from equation 7.11;
end
Obtain the sequence Expected Free Energy $\sum_t^{\mathcal{T}} \mathrm{G}(\pi, \tau)$ following equation 7.8;
Select best action sequence $\pi \leftarrow min_\pi R(\pi)$;

7.6 Experiments

We assess the behaviour of an agent in the *Flappy Bird* environment (fig. 7.2). This is a task where a bird must navigate between obstacles (pipes) at different positions while stabilising its flight. Despite the apparent simplicity, the environment offers a fundamental aspect present in the physical world. Namely, the inherent dynamics lead spontaneously to the termination of the agent in the environment. The biological analogy corresponds to the functional disintegration of the agent and the inability to maintain their essential variables within bounds. If the agent stops propelling, it succumbs to gravity and falls. The environment also has a constant scrolling rate, which implies that the agent cannot remain floating at a single point and cannot survive simply by flying aimlessly.

Originally, the task provides a reward every time the bird traverses between two pipes. However, for our set-up, the information about the rewards is never transmitted to the agent and therefore does not have any impact on its behaviour.

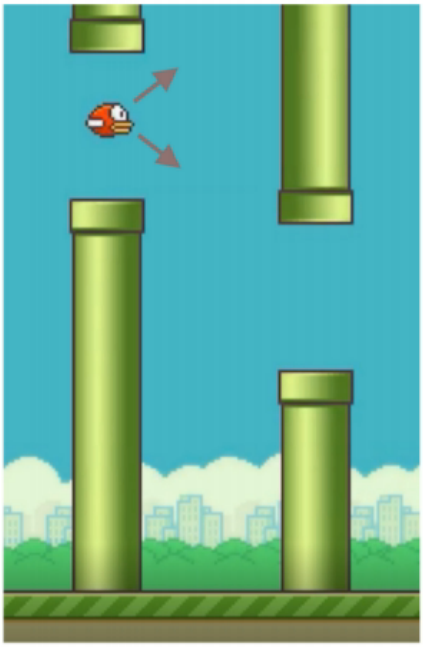


Figure 7.2: *The Flappy Bird environment.*

The agent receives a feature vector of observations indicating its location and those of the obstacles. In addition, the agent obtains a *measurement* v indicating its presence in the task (i.e. 1 or 0). We again emphasise that this measurement does not represent anything positive or negative by itself, it is simply another signal that we assume the agent can calculate. The measurement, as elaborated in the previous section, represents a coarse-grained internal state that acquires significance according to the agent-environment coupling and the imperative to minimise free energy that is assumed in active inference (figure 7.3).

Similarly to the outline in 7.3, the agent monitors the last $t - k$ values of this measurement and estimates the density to obtain θ_t. These become the statistics describing the current approximated distribution of preferences $p(v|\theta_t)$ or $p_{\theta_t}(v)$. These θ are also used to augment the observations to $x_t = (o_t, \theta_t)$. When the agent takes a new measurement v_t, it evaluates the surprisal against $p_{\theta_{t-1}}(v_t)$. For this experiment because v is a binary value we can evaluate it with a Bernoulli

Figure 7.3: *The agent obtains a measurement v_1 related to its viability in the current environment. Initially, the agent cannot interpret this measurement but as time passes it learns to associate the signal to a specific internal state.*

density function as,

$$-\log p_{\theta_{t-1}}(v_t) = -\Big(v_t \log \theta_{t-1} + (1 - v_t)\log(1 - \theta_{t-1})\Big) \tag{7.12}$$

First, we train a baseline model-free surprisal minimising DQN as specified in 7.3 and parameterised by a neural network. Then we examine the behaviour of a second agent that minimises the EFE. For the EFE agent, it learns an augmented state transition model of the world, parameterised by an ensemble of Neural Network (NN), and an expected surprisal model, also parameterised by another NN (appendix E). RHE is used to generate and identify a planning action-sequence as described in 7.5.1.

7.6.1 Simulation results

The plot in fig. 7.4 tracks the performance of an EFE agent in the environment (averaged over 10 seeds). The dotted line represents the surprisal minimising DQN agent after 1000 episodes. The left axis corresponds to the (unobserved) task reward while the right axis indicates the approximated number of time steps the agent survives. During the first trials, and before the agent exhibited any form of competence, it was observed that the natural coupling between the agent and environment grants the agent a life expectancy of approximately 19-62 time steps

Figure 7.4: *Performance of an EFE agent. The left axis indicates the unobserved rewards as reported by the task and the right axis is the number of time steps it survives in the environment. The dotted line shows the average performance of an SM-DQN after 1000 episodes.*

in the task. This is essential as it starts to populate the statistics of v. Measuring a specific quantity v, although initially representing just another signal, begins to acquire a particular value due to the frequency it occurs. In turn, this starts to dictate the preferences of the agent as it hints that measuring certain signals correlates with having a stable configuration for this particular environment as implied by its low surprisal.

Figure 7.5 shows the evolution of parameter θ (averaged within an episode) corresponding to the distribution of preferred measurements $p_\theta(v)$ which determines the level of surprisal assigned when receiving the next v. As the agent reduces its uncertainty about the environment it becomes more capable of associating sensorimotor events with specific measurements. The behaviour becomes more consistent with seeking less surprising measurements, and as we observe, this reinforces its preferences, exhibiting the circular self-evidencing dynamics that characterise an agent minimising its free energy.

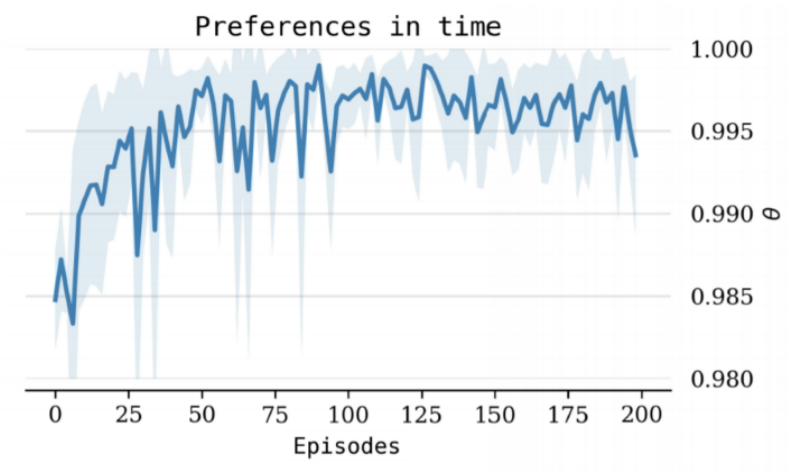


Figure 7.5: *Parameter θ in time, summarising the intra-episode sufficient statistics of $p_\theta(v)$.*

7.7 Discussion

The main premise in active inference is the notion of an agent acting to minimise its expected surprise. This implies that the agent will exhibit a tendency to seek the sort of outcomes that have high prior probability according to a biased model of the world, giving rise to goal-directed behaviour. In this chapter, we have explored the emergence of this biased model that captures the preferences of an agent. We constructed a scheme inspired by organismic-based considerations of viability and various insights from the literature on the nature of agency and adaptive behaviour (von Uexküll, 1926; Maturana and Varela, 1987; Weber and Varela, 2002; Di Paolo, 2003, 2005, 2010; Damasio, 2004; Barandiaran et al., 2009; Polani, 2009; Salge and Guckelsberger, 2016; Man and Damasio, 2019). Collectively, these views suggest that the inherent conditions of precariousness and the perturbations an agent must face are crucial ingredients for the emergence of purpose-generating mechanisms. In that sense, our main concern has been to explore an instance of the conditions in which a stable set of attracting states arises, conferring value to observations and leading to what seemed like self-sustaining dynamics even when all measurements lacked any initial functional value.

However, the scheme is far from being general and serves as a proof-of concept. Two main difficulties arise in its application to other domains. First, we introduce a variable associated to the agent's *internal state*, which we connect to its operational integrity. Second, we manually determine a separation between internal and external observations and define the internal variables as the target. For the first point, it is currently not well studied how to establish a principled protocols to define internal states and variables in an agent. For example, in research on homeostatic RL, the internal variables of interest are predefined similar to what was presented in this chapter (Keramati and Gutkin, 2011; Cos et al., 2013; Laurençon et al., 2021). The second point relates also to the discussion in the previous chapter regarding the more basic problem of relevance, which is discovering autonomously the target variable.

Agent-Environment coupling: A matter of further analysis, is the role of the environment to provide structure to the behaviour of the agent. In the work presented here the initial set of internal measurements afforded by the environment contributes to the formation of a steady state where the visual features inform the actions necessary to maintain the desired internal measurements. Hence, the initial conditions of the agent-environment coupling that furnish the distribution $p(v)$ provide a starting solution for the problem of self-maintenance as long as the agent can preserve the statistics. Thus we could establish that when the agent either lacks a sophisticated sensory apparatus or the capacity to extract invariances from external stimuli, then tracking its internal configuration may suffice for some situations. However, this requires further unpacking. Not only because as discussed earlier it remains uncertain how to define the internal aspects of an agent, but also because often simulations and tasks do not capture the essential characteristics of real environments either.

Drive decomposition: While here we have afforded our model certain levels of independence between the sensory data and the internal measurements, it might be sensible to imagine that internal states would affect perception and perceptual

misrepresentation would affect internal states. Moreover, an agent should move from objectives based entirely on survival to acquire other higher-level goals and preferences. For that to occur it would have to learn to integrate and balance these multi-level objectives. From equation C.1 we can conceive a simplified scenario and establish the following expression (appendix F),

$$\begin{aligned} \mathrm{G}(\pi,\tau) \approx & \underbrace{\mathbb{E}_{q(o_\tau,v_\tau,\theta,s_\tau|\pi)}[\ln q(s_\tau|\pi) - \ln p(s_\tau|o_\tau,\pi)]}_{\textit{Epistemic value}} \\ & \underbrace{- \mathbb{E}_{q(o_\tau,v_\tau,\theta,s_\tau|\pi)}[\ln p(o_\tau)]}_{\textit{High level value}} \\ & \underbrace{+ \mathbb{E}_{q(o_\tau,s_\tau|\pi)} H[p(v_\tau|s_\tau,o_\tau,\pi)]}_{\textit{Regulatory value}} \end{aligned} \tag{7.13}$$

Where the goal-seeking value is decomposed into a component that considers preferences encoded in a distribution $p(o)$ and another element estimating the expected entropy of the distribution of essential variables. With this expression, policies should balance the different contributions and resolve hypothetical scenarios, such as the case of a higher-level goal being at odds with the viability of the system.

8 Discussion

Research in model-based agents has been bolstered on one hand by advancements in different fronts of machine learning, and on the other by potential benefits in sample efficiency, generalisation, or interpretability. For the author, the context which motivated this research was due to our aim to integrate the world-modelling techniques of the time with existing planning algorithms. We initially found that this integration was not as trivial as expected due to catastrophic compounding errors and the lack of architectures capable of modelling stochastic dynamics. Influenced by the ideas of the time, our first attempt was intended to try to increase the reconstruction accuracy. Our perspective on how to deal with or think of the inaccuracies in the models naturally evolved and converged in the notions presented in chapter 6 and others that we will discuss in this chapter. The results of our efforts can be summarised as follows:

- We introduced an error-correction scheme based on ensembles. This allowed us first, to get a tangible sense of how uncertain were the models about the future due to the difference in their predictions. Second, to allow those multiple predictions to dictate how to correct them leading to improved accuracy and better agent performance.
- Given a picture of forward models where uncertainty can be reduced but will ultimately remain an integral part of the model, then the question is how to design an architecture that accommodates it rather than trying to ignore it. We capitalised on advancements in sequential latent variable models to account for the uncertainty in the state transitions. We designed

an agent conferred with a state-space model to learn to simulate external dynamics and with the capacity to plan in latent space, thus introducing a latent variation of Rolling Horizon Evolution (RHE) for approximate forward models. This could be considered as a discrete-action counterpart of Hafner et al. (2019).

- We then extended the previous agent to also acquire a forward model of the Q-value transitions. Different from other proposals in model-based RL where a forward model is used to simulate experience to train and estimate the Q-values but does not learn its forward model, here the agent acquires it and uses it to explicitly evaluate candidate action sequences via RHE.
- The role of uncertainty in world modelling then took a different turn as we later reflected upon the pursuit for accuracy and whether a forward model should aim to match the environmental complexity. If an agent is not capable of approximating it, either due to the environment's complexity or due to a limitation in the agent's perceptual resources, then on what aspects should the agent focus its efforts? What should it prioritise? And how should we include these limitations as part of the problem specification? Preliminarily, we decided to tackle these issues by adapting insights from the information theory literature. We took *predictive information* (Bialek and Tishby, 1999), reframed as an information bottleneck (Tishby et al., 2000; Still, 2014), as a basis to formulate an agent-centric form of predictive information. The expression that we proposed encourages the agent to store essential information (e.g. its latent state) about what it predicts will also be essential in the future (e.g. future latent state). The prediction is confined to the same agent's latent space. This is in contrast to a reconstructive model that encourages the storage of information that is essential for the reconstruction of an observation despite that some of the information ceases to be essential in the future. Besides the expression, we also introduced a practical training scheme by approximating mutual information bounds and

then integrated the novel model-learning objective with the RHE planner.

- We ended with questions about how a model can take a role that extends beyond the prediction of the future to actively influence the behaviour of the agent. Research in theoretical neuroscience has precipitated in the past years towards an understanding of an agent and its interactions in terms of probabilistic and information-theoretic constructs. A framework that has tried to unify some of these views is active inference which in its most essential form could be interpreted as a formalisation for homeostatic behaviour. Central to its formulation is the existence of a reference distribution that encodes the preferences or desires of the agent. Although it has been argued in the literature that this distribution is refined over evolutionary scales and during the life span of the agent, it has not been shown clearly how this distribution could arise in a self-supervised manner. Examples in the literature predefined this desired distribution. We provided a simple example of how an instance of this distribution could emerge as a consequence of the statistics of the agent-environment coupling. The statistics, which are initially spontaneous and void of significance for the agent, start to acquire a normative character which leads to their self-reinforcement. Within the context of the thesis then we observed how the uncertainty in the forward model (or more precisely in this case, in a forward sub-model) can dictate the value of the signals and therefore actively influence the behaviour of the agent.

8.1 Outlook, limitations and future work

8.1.1 To learn or not to learn a model?

We presented motivating arguments for approaching sequential decision-making problems with model-based approaches as opposed to model-free tools. Let us

consider how these arguments translated into practice during our experience working on model-based methods.

Initially, the most noticeable aspect was the trade-off between the number of samples necessary to train a model and the time required to train it. In our experience, model-based agents needed fewer steps to achieve some form of competence in the tasks. However, this came at a significant increase in time and computational requirements. The decision to construct a model entails the inclusion of several additional components compared to model-free agents. For example, the architectures from chapter 5 required Recurrent Neural Network (RNN), Variational Auto Encoder (VAE), and the planner in addition to learning Q-values. The mixture of these extra components also brings other types of consequences. Each of these components comes with its own set of hyperparameters, such as layer size, learning rates, decay rates, training regime, number of rollouts, mutation rate, and so on. The process of training the architecture becomes more complex and time-consuming. Not only because of the explosion of the hyperparameter space search but also due to known instabilities in training Reinforcement Learning (RL) agents (Mnih et al., 2015; Henderson et al., 2019; Dulac-Arnold et al., 2021). A task which increased considerably in complexity if we consider that we have to train the other components simultaneously. Although we made early tests in other environments such as various Atari games or Super Mario, these factors were eventually major impediments that led us to opt for simpler and more controlled environments.

We did not explore in detail the topic of generalisation or transferability, however, we tested our architectures in procedurally generated Sokoban and Gridworld environments. There are some observations we noticed on the subject from our unpublished results. Transferability does not come for free. We observed that in some cases it was favourable to start with a previously learned model. However, the predictions generated by the model when applied to a new instance of the task were too entangled. The architecture had to invest time in unlearning aspects

of the model it had acquired before, often making predictions that combined dynamics from several instances of the task. We speculate that the source of the problem is that predictions are not compartmentalised, the environment is predicted as a whole since that is what is encouraged by the loss function during training. Potentially research in object discovery and object-centric representations could help to avoid some of these effects (Watters et al., 2019; Yoon et al., 2023).

Nonetheless, this capacity to look at what the model is predicting and being able to diagnose potential problems is certainly appealing for interpretability. Ultimately, the answer to whether to learn or not to learn a model is, at least for the moment, tied to the particular context we are interested in solving or the information we have about the problem. Similar to how practitioners apply control-theoretic or planning methods when a model is already available. It is certainly true that model-free methods are still currently state-of-the-art in many common benchmarks (Fan et al., 2022; Fan and Xiao, 2022) and have been more amenable to integrating complementary extensions (Hessel et al., 2017; Badia et al., 2020). This could also be because there is a higher volume of research and accessibility to model-free methods compared to model-based ones. The future outlook, however, might be that the line between model-free and model-based could be blurrier than initially thought (Dayan, 1993; Janner et al., 2021; Eysenbach et al., 2022; Ghugare et al., 2023).

8.1.2 On rewards

An aspect where learning a model might not just be advisable but also required is in many principles formulated in the intrinsic motivation literature. In chapters 5 and 6 we noted that there are stages in Minipacman in which the agent is confined to an area of the maze seemingly getting stuck. We speculated that because the surroundings did no longer have edible cells the task effectively became a sparse

reward task. In this scenario adding some form of *intrinsic* reward to the plans could guide the agent to explore other regions. Considering that the architecture learns next state transitions, there are intrinsic motivation principles that could be conveniently adapted to extend the architecture. For example, predictive information, which we reviewed in chapter 6 has been applied in robots and artificial agents to encourage the emergence of complex dynamics (Martius et al., 2013, 2014). Another example is the introduction of an information gain term. This can take different forms. For instance, as the expected amount of change in a model (Little and Sommer, 2013), a state transition, or as the cost of action selection (Tishby and Polani, 2011; Rubin et al., 2012) if observing an outcome.

8.1.3 What to learn?

In our work, we have let the gradients flow unrestricted throughout the whole architecture. At first glance, this could appear as nothing more than a trivial technical decision. Often in publications, there is no mention of whether the gradients flow unrestricted and this only becomes apparent if the source code is publicly available. In other cases, it is mentioned that representation learning is kept separated from control or task learning although the rationale behind this decision is not provided. Outside of model-based RL, there has been debate on whether perception and representation learning should be learned separately from policy learning. A reason to proceed in this manner is that there can be a discrepancy between the complexity required to learn a policy and the one required for extracting visual features (Cuccu et al., 2019). It is argued that in many cases, the policy required to achieve a task can be relatively simple, but because the architecture has to deal with the larger complexity of learning higher-level features it might interfere with the acquisition of a good policy. Thus having a dedicated component to focus on the policy could lead to improved performance. Another part of the issue is that the feedback provided from the

reward signal can be insufficient to learn adequately both problems (Stooke et al., 2021). It remains unknown to what extent these arguments hold, for example, in the model-based setting where the prediction of additional targets provides extra sources of supervision. Prominent architectures in model-based RL such as those proposed in World Models (Ha and Schmidhuber, 2018), Dreamer (Hafner et al., 2020, 2022) or SLAC (Lee et al., 2020) have maintained the separation between the perceptual component and the controller. Here, although we made the deliberate decision to let the gradients flow freely to influence the representations, We did not find that it affected the stability during training. On the contrary, we found that it increased the performance in the tasks.

Recently, there have been theoretical arguments in RL that outline some of the properties of value-based forward models such as those learned by MuZero. The main argument is that there exists a class of models that are *value-equivalent* for a space of policies and value functions if all of the models within the class produce the same updated value function (Grimm et al., 2020, 2021). In other words, there are agents with different models of the dynamics of the environment but with equivalent performance. Thus, for example, it could be possible that there might be agents with non-reconstructive forward models that produce the same value estimation as an agent with a perfect model of the environment. In a pragmatic sense, we have seen that purely value-based models such as the Value Prediction Network or MuZero have effectively bypassed the need to reconstruct the environment by relating the representations directly to aspects of control. However, these have also been shown to be less sample efficient compared to reconstructive architectures (Schrittwieser et al., 2020; Hafner et al., 2022).

There will likely be technical improvements that will accelerate the way we learn these control-related targets. Alternatively, we can also speculate that there exists additional targets or constraints to encourage representations with an increased amount of actionable information from the environment to facilitate decision-making. We have attempted to operationalise this intuition in chapter

6 where we showed that in some scenarios an agent whose representations aim to predict their future latent states could be more advantageous than an agent whose objective is to represent the external environment. In the next section, we elaborate further on how to refine and expand this idea.

8.1.4 Targets for non-reconstructive models

Let us make a few speculations about potential refinements to the ideas developed in chapter 6. As we have seen, an agent with an imperative to learn representations of the world for its own sake has to maintain the information that is necessary for the reconstruction of external observations. On the other hand, we could alternatively interpret the flow of sensory data as an opportunity for the agent to collect information to determine potential actions rather than for reconstruction. The degrees of freedom of the agent's behavioural repertoire cannot match the degrees of freedom and complexity of the environment (Ashby, 1958). Therefore, it is reasonable to assume that there is less incoming information that is necessary for the agent to decide than to reconstruct a snapshot of the environment. As a starting point, we could then define, for example, the following information bottleneck:

$$\min_{q(z|x)} [I(X_t; Z_t) - I(Z_t; A_t)]$$

which expresses that the representation Z_t at time t is going to limit the amount of information that is extracted from the sensory stream X_t while maintaining that which is maximally informative for action A_t. This type of strategy is reminiscent of information-theoretic approaches of the perception-action loop where a trade-off determines a relationship between the sensory stream and action (Polani et al.,

2006; Still, 2009).[1,2]

We can also extend this analysis to the realm of predictive information. In chapter 6 we noted that predictive information as it is, is formulated as a measure of the structural complexity of a process rather than from the agent-centric view that we require. If only a fraction of the data from the past is required to predict future data then we can also assume that most of the future data is irrelevant to the prediction of subsequent futures. Moreover, for an agent, the presumption of predictive information, as it is, would be again to consider that its objective is the prediction of the future rather than the prediction of the part of the future that is relevant to its decision-making. We considered then that the latent future factors could be used to constrain a part of the future that is significant for the agent. However, we noted in the discussion that the selection of these relevant variables is an open question. Some analyses have recognised that adaptive value, by which we mean factors that are beneficial or detrimental for an agent, plays a crucial role in determining what are the relevant variables within a particular context (Bialek et al., 2006; Taylor et al., 2007; Kolchinsky and Wolpert, 2018; Pedraza et al., 2018). Moreover, adaptive value may serve as a criterion to bridge predictive capacity and optimal performance (Bialek et al., 2006). This suggests that we could extend our analysis to connect it with utility and decision-making. For instance, we could modify the information bottleneck in eq. 6.9 to connect

[1] However, in Polani et al. (2006) the rate-distortion tradeoff $\min_\pi I(X_t;A_t) - \mathbb{E}[V(X_t,A_t)]$ does not explicitly include an intermediate representation Z_t. Instead, the action A_t itself acts as a form of representation. To select a potential action an agent must limit the amount of information it takes from the sensory stream X_t. The second term of the tradeoff instructs the agent to select the action that maximizes the expected value at a particular state. This effectively connects the information extracted from the environment to the attainment of an incentive, which we have not elaborated on at this moment.

[2] In the case of Still et al. (2012) they define a predictive information tradeoff $\max_{\pi,p(z|h)}[I(\{Z_t,A_t\};X_{t+1}) - I(Z_t;H) - I(A_t,H)]$ where there is access to a history H of past observations of arbitrary size. As we know from predictive information, the representation Z_t discards as much information from the past while maximising what it can predict about the future. In this case the future observations X_{t+1}. Similar to Polani et al. (2006) the agent aims to extract as little information as possible from the past sensory stream to take an action. However, here the tradeoff is not connected directly to a utility (e.g. value maximisation) beyond the prediction of the future.

more explicitly the representations to the value of future potential actions:

$$\min_{q(z_{t:t+1}|o_{t:t+1})} [I(\{O_t, Z_{t-1}, A_{t-1}\}; Z_t) - I(\{Z_t, A_t\}; Q_{t+1})] \tag{8.1}$$

this expression then indicates the following: an agent should aim to store and represent the information from the past and present that is going to be relevant in the future towards the prediction of Q-values, which in turn, help to guide its action selection. The previously described architectures of Value Prediction Networks (Oh et al., 2017) and MuZero (Schrittwieser et al., 2020) could be considered as precedents close to this treatment. As we briefly described, these architectures predict the current value and policy from observations without trying to reconstruct them, thus we could interpret them as implicit bottlenecks of the present that estimate the value and policy for the current time step. Nonetheless, besides being deterministic, these architectures follow a greatly different learning scheme and structure compared to the strategies pursued in this thesis. Most crucially, however, their forward model is not oriented towards forms of predictive information. This is in the sense that the value and policy estimates at time $t + k$ are based on the actual or simulated embedding also at time $t + k$. This is in contrast, for instance, to the prediction of a value at $t + k + 1$ from an embedding at $t + k$. Another closely related approach is the previously mentioned rate-distortion trade-off between information and expected utility proposed by Polani et al. (2001, 2006). The objective encourages an agent to learn policies that commit to using the least amount of information to attain an expected amount of utility and can be optimised as a single free energy functional to solve Markov Decision Process (MDP) (Tishby and Polani, 2011; Rubin et al., 2012). In our case, our hypothetical objective is not concerned with learning a policy directly but rather with acquiring representations that are then used for online planning.

8.1.5 Goal, Drives, and Motivations

Another crucial aspect that determines what is relevant at a given moment is the goal or the objective of an agent. In chapter 7 we had a small incursion away from the classic specification of exogenous rewards and the reward maximisation scheme from RL. However, in this thesis, we have largely ignored this element of behaviour. It can be argued that reward maximisation encodes implicitly a drive and some form of proto-intentionality, but it offers little interpretability regarding the skills and goals an agent may have. Moreover, reward misspecification can lead to well-known problems in RL such as reward exploitation or sparse reward tasks. For active inference schemes, we have seen how the notion of a distribution of preferences can be a more expressive alternative to a scalar reward, but in previous work, this distribution has also been predefined externally. We showed a proof-of-concept of how it could emerge and evolve in a self-supervised manner from the statistics of the agent-environment coupling. However, without further research into a formal notion of the internal state of an agent, it will remain challenging to scale to more complex scenarios.

We could then take an intermediate approach to relate the effects of an agent's goals to the formation of representations and forward models. Goal conditioned RL (Schaul et al., 2015; Andrychowicz et al., 2018), augments the RL tuple with the inclusion of an additional variable g that defines the current goal. Then it becomes possible to express $Q(s, a; g)$ as the expected utility that an agent can obtain if it takes action a from a state s according to a goal g. Although it is still necessary to define metrics to obtain pseudo-rewards, the opportunity to contextualise the value of a state-action pair depending on a goal offers increased flexibility. From the perspective of the themes that concern us in this thesis, the goal could act as an additional constraint to prevent the architecture from learning representations that merely try to map external observations and instead encourage it to extract information that is relevant to the behavioural needs of

the agent. From what we have discussed in the previous section we can conceive two hypothetical scenarios to explore in the future. The first is the extension of chapters 5 and 6. Similar to the way that an action informs the architecture that a transition must be learned for a specific action, the inclusion of a goal in a Q-function provides extra information about the particular kind of transition we want to learn. Therefore this could result in Q-value models learned for outcomes that are more compatible with the agent's objectives. The second scenario follows from eq. 8.1 where Q_{t+1} would correspond to a goal conditioned Q-value. The representation Z_t would then capture information that is relevant for the prediction of Q-values as determined by the agent's goals rather than for any future Q-value.

8.1.6 Architecture enhancements

From a technical standpoint, there are also several improvements that we could apply to the architectures. The ensemble approach used in chapters 3 and 7 could be extended to include the State Space Model (SSM) architectures. It could allow us not only to consider the model uncertainty of the state transitions or the Q-values but also to simulate the variance of future time steps. Other promising advancements have been recently published during the completion of this thesis. For example, Micheli et al. (2022) presented an architecture that learns environment dynamics using a Transformer (Vaswani et al., 2017), a neural network module commonly used in language tasks. It considers that the image tokens generated by an autoencoder can be considered part of a language. In our architectures, the Transformer would substitute the RNN to process temporal dependencies. Unlike RNNs, Transformers can process every step of a sequence simultaneously which facilitates the discovery of intricate relationships among multiple steps which has been said to resemble a form of attention. In the specific architecture in Micheli et al. (2022) the learned model is used by the agent to

learn a policy in latent space similar to Dreamer (Hafner et al., 2020, 2022) and SLAC (Lee et al., 2020) but not for online planning. Integrating it within our architecture would allow the learned models to also be used to simulate planning sequences.

8.2 Final words

Our central concern in this thesis has been how to plan and act despite the presence of uncertain dynamics reflected in an agent's internal model. The theme has converged towards the necessity to think about how these limitations could be a fundamental part of the design of perceptual, learning, and behavioural strategies. The question is broad, and the themes explored in this thesis are merely a few of a multitude of factors to consider for the integration between world modelling and decision-making. The task is an all-encompassing cognitive problem that spans a wide array of functions and mechanisms ranging from memory to attention and categorisation. It entails a series of open questions, challenges, and technical breakthroughs, as well as a deeper conceptual understanding of the perception-action loop to provide a refinement or a new perspective in the formulation of the elements involved in the interaction between an agent and the environment.

A Preliminaries

A.1 Reinforcement Learning

RL is a normative paradigm concerned with the process of training an agent through trial and error with the intention that the agent learns to become competent at a task. Inspired by the behavioural psychology literature, central to the formulation of RL is the notion of reward (i.e. and punishment as a negative reward), that serves to encourage or discourage certain behaviours.

The RL framework is formalised in the following manner. Consider an agent that interacts with an environment in discrete time. At every time step t the agent observes a state $s_t \in \mathcal{S}$ and selects an action $a_t \in \mathcal{A}$ according to a policy $\pi \in \Pi$. Depending on the case this policy might be stochastic and defines a distribution over actions thus $a_t \sim \pi(a|s)$. Once the environment registers the action selected by the agent it emits a new observation s_{t+1} and a reward $r_{t+1} \in \mathcal{R}$.

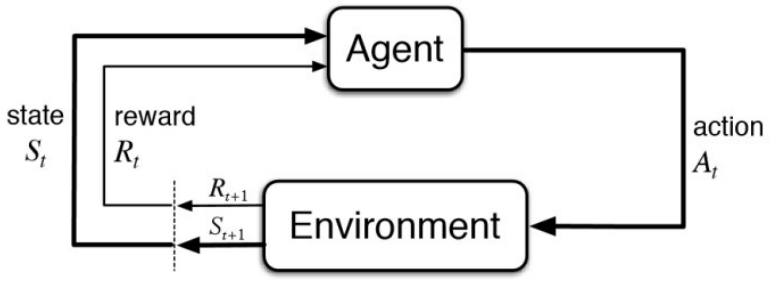


Figure A.1: *The reinforcement learning agent-environment loop.*

RL operates under the assumption of the reward hypothesis (Sutton and Barto, 1998). The objective is to acquire an action policy that maximises the expected

cumulative reward

$$G_t = \sum_{k}^{\infty} \gamma^k r_{t+k} \tag{A.1}$$

also known as the return G. Here γ is a discount factor that makes the sum well-defined for infinite sums and determines the importance of immediate rewards in relation to distant rewards. Historically, RL methods have fallen into three major different categories:

- Value-based methods (Watkins and Dayan, 1992; Rummery and Niranjan, 1994) that find approximate solutions to the Bellman equation (Bellman, 1952) through the estimates of the value of a state.
- Policy gradient methods (Williams, 1992) that operate directly in the space of policies by searching for parameters that improve the performance of the policies.
- Actor-critic methods (Konda and Tsitsiklis, 1999) that combine both approaches, by using the state-value estimates to guide the policy search.

For the context of the thesis, we limit our review to the value-based approach in the next section.

A.1.1 Value-based methods

These methods depend on the accurate estimation of state-value functions to inform how the agent should act in specific states. The state-value functions denote the expected return an agent would obtain by following a policy π from a state s and can be formulated as

$$V^{\pi}(s) \triangleq \mathbb{E}_{\pi}\left[\sum_{k}^{K} \gamma^k r_{t+k} \middle| s = s_t \right] \tag{A.2}$$

In addition, one can also define the action-value function Q to determine the expected return from taking a specific action

$$Q^{\pi}(s,a) \triangleq \mathbb{E}_{\pi}\left[\sum_{k}^{K} \gamma^{k} r_{t+k} | s = s_t, a = a_t\right] \tag{A.3}$$

which can be decomposed by expressing it in the recursive Bellman expectation equation form

$$Q^{\pi}(s,a) = \mathbb{E}_{\pi}[r_t + \gamma V^{\pi}(s_{t+1})] \tag{A.4}$$

the objective for the agent can be summarised in two parts, it has to attempt to find the optimal Q-function over policies

$$Q^{\star}(s,a) = \max_{\pi} Q^{\pi}(s,a) \tag{A.5}$$

and act according to the optimal policy based on the optimal Q-function

$$\pi^{\star}(a|s) = \text{argmax}_{a} Q^{\star}(s,a) \tag{A.6}$$

A.1.2 Model-Based Reinforcement Learning

Thus far there has been no explicit mention of the transition function that determines the underlying dynamics of the environment. The previous section described an agent that remains agnostic to it and derives its behaviour from the estimates supported by the value functions. Although it can be suggested that the value functions encapsulate a form of rudimentary knowledge of the future, the agent is *model-free*. That is, the agent is reactive and acts given only the present, as it does not explicitly simulate the future to assess hypothetical scenarios.

Let us first consider an equivalent way to specify equation A.4

$$Q^{\pi}(s,a) = \sum_{s'} \mathrm{T}^{a}_{ss'} \left[\mathrm{R}^{a}_{s} + \gamma \sum_{s'} \pi(a'|s') Q^{\pi}(s',a') \right] \tag{A.7}$$

Here $\mathrm{T}^{a}_{ss'} = p(s_{t+1} = s' | a_t = a, s_t = s)$ is the transition function that maps the probability of going from state s taking action a to a next state s'. $\mathrm{R}^{a}_{s} = \mathbb{E}[r|s,a,s']$ is the expected reward received when transitioning from s taking action a to s'. What the equation tells us is that while the value functions depend on the laws of the environment, the agent does not have to know what these are. The agent is only concerned with estimating the value functions given the samples it has gathered during the interaction with the environment.

As we began to motivate in the introductory chapter, we are particularly interested in the advantages and the foresight that an agent potentially gains with a model of the environment. Thus an agent has to learn to estimate the conditional $p(s_{t+1}|s_t, a_t)$ and more concretely

$$p(s_0, a_0, ...s_T, a_T) = p(s_0) \prod_{t}^{T} \pi(a_t|s_t) p(s_{t+1}|s_t, a_t) \tag{A.8}$$

we can consider a trajectory τ as a sequence of states and actions $\tau = \{s_0, a_0, ..., s_T, a_T\}$. Therefore the agent must learn to simulate the transitions that occur up to the time horizon T in the future. With this model, the agent should be able to simulate potential outcomes and find a policy

$$\pi = \mathrm{argmax}_{\pi} \mathbb{E}_{\tau \sim p(\tau)} \left[\sum_{t} r(s_t, a_t) \right] \tag{A.9}$$

that maximises the expected sum of rewards according to the forward model.

A.2 Information Theory

A.2.1 Self-Information

The self-information, also known as *information content*, *surprise*, or *surprisal*, measures the amount of information gained from the observation of a single event or outcome $x \in X$.

$$I(x) = -\log p(x) \tag{A.10}$$

the less probable an event the larger the self-information, and accordingly, an event that is highly probable provides a low information content.

A.2.2 Entropy

The Shannon entropy is the expected value of self-information of a random variable X. It is defined as

$$\begin{aligned} H(X) &= -\sum p(x) \log p(x) \\ &= \mathbb{E}_{p(x)}[I(x)] \end{aligned} \tag{A.11}$$

intuitively, entropy quantifies the level of uncertainty about the outcome of X. That is, entropy measures the amount of uncertainty of the entire distribution. The more uniform a distribution is, the higher the entropy and therefore the larger the reduction in uncertainty once an outcome is observed.

A.2.3 Conditional Entropy

The conditional entropy considers the inclusion of another random variable Y and measures the amount of uncertainty that remains in X if Y is known

$$H(X|Y) = -\sum_{y} p(y) \sum_{x} p(x|y) \log p(x|y) \tag{A.12}$$

A.2.4 Kullback-Leibler Divergence

The Kullback-Leibler (KL) divergence measures how different two distributions p and q are

$$D_{KL}[p||q] = \sum p(x) \log \frac{p(x)}{q(x)} \tag{A.13}$$

the KL divergence is widely used in machine learning because often it is necessary to approximate an intractable distribution p with another distribution q. The KL divergence admits various interpretations. A way to think about it is a measure of the amount of information lost if approximating p via a particular distribution q.

Formally, the KL is a divergence and not a true metric. It cannot measure the distance between q and p because it is not symmetric, and thus generally $D_{KL}[p||q] \neq D_{KL}[q||p]$, as they entail two different forms of approximating p. $D_{KL}[p||q]$ is known as the forward KL divergence, and using it as optimisation objective, it exhibits *mean seeking*, *zero avoiding* and *inclusive* behaviour. While the reverse KL divergence $D_{KL}[q||p]$ is *mode seeking*, *zero forcing*, and *exclusive*.

A.2.5 Mutual Information

It was mentioned that the conditional entropy $H(X|Y)$ accounts for the remaining uncertainty in X if Y is known. If instead, the intention is to measure the expected

amount of information that is gained from X by observing Y, this can be calculated with the mutual information

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X) \tag{A.14}$$

therefore high mutual information denotes a large reduction in uncertainty. It can also be expressed as

$$I(X;Y) = D_{KL}[p(X,Y)||p(Y)p(Y)] \tag{A.15}$$

in this form, it can be observed more clearly that the mutual information is zero if X and Y are independent. That is, observing X or Y does not reduce the uncertainty of the other.

A.3 Variational Inference

The major idea behind Bayesian statistical inference is to learn about unobserved variables from observed data and reason about their uncertainty. Assume z is the hidden variable of interest and x is the observed data. Inferring z from x can be done through Bayes's theorem

$$\begin{aligned} p(z|x) &= \frac{p(x,z)}{p(x)} \\ &= \frac{p(x|z)p(z)}{\int p(x|z)p(z)dz} \end{aligned} \tag{A.16}$$

The main obstacle in estimating this expression lies in the marginal density $p(x)$ which tends to be a high-dimensional and intractable integral. As a consequence,

exact inference is for most situations unattainable and therefore approximation methods must be used to estimate the posterior $p(z|x)$. Among the most popular include Markov Chain Monte-Carlo (MCMC) (Metropolis et al., 1953) and other MCMC-based sampling methods such as the Gibbs sampler (Geman and Geman, 1984) or Hamiltonian Monte Carlo (Duane et al., 1987). MCMC schemes are unbiased, therefore they converge to the true posterior in time. However, this process can also be slow. An alternative approach is to recast the process of inference as an optimisation problem through the principles of variational inference. The optimisation objective can be expressed in the following way

$$q^{\star}(z|x) = \underset{q(z|x)\in \mathrm{Q}}{\operatorname{argmin}} D_{KL}\left[q(z|x)||p(z|x)\right] \tag{A.17}$$

thus the intention is to find a simpler proposal distribution $q(z|x)$ (often simply denoted as $q(z)$) that approximates the true posterior $p(z|x)$ as close as possible. Decomposing the $D_{KL}[q(z|x)||p(z|x)]$ we obtain

$$\begin{aligned}
D_{KL}[q(z|x)||p(z|x)] &= \int q(z|x) \log \frac{q(z|x)}{p(z|x)} dz \\
&= \int q(z|x) \log \frac{q(z|x)p(x)}{p(z,x)} dz \\
&= \int q(z|x) \log \frac{q(z|x)p(x)}{p(z,x)} dz \\
&= \int q(z|x) [\log q(z|x)p(x) - \log p(z,x)] dz \\
&= \int q(z|x) [\log q(z|x) + \log p(x) - \log p(z,x)] dz \\
&= \int q(z|x) \left[\frac{\log q(z|x)}{p(z,x)} + \log p(x)\right] dz \\
&= D_{KL}[q(z|x)||p(z,x)] + \log p(x)
\end{aligned} \tag{A.18}$$

which as expected, shows that for the objective to be $D_{KL}[q(z|x)||p(z|x)] = 0$ it is

necessary to take into account the intractable marginal $p(x)$. For the moment, let us concentrate instead on the first term which minimises the divergence between the variational posterior $q(z|x)$ and the joint $p(z,x)$. This can be decomposed into

$$\begin{aligned} D_{KL}[q(z|x)||p(z,x)] &= \int q(z|x) \log \frac{q(z|x)}{p(z,x)} dz \\ &= \int q(z|x) \log \frac{q(z|x)}{p(x|z)p(z)} dz \\ &= \int q(z|x) [\log q(z|x) - \log p(x|z)p(z)] dz \\ &= \int q(z|x) [\log q(z|x) - \log p(x|z) - \log p(z)] dz \end{aligned} \tag{A.19}$$

if instead of minimising this expression we perform the equivalent operation of maximising it, we can refer to it in statistics as the Evidence Lower Bound (ELBO), which can be decomposed as

$$-D_{KL}[q(z|x)||p(z,x)] = -\int q(z|x) [\log q(z|x) - \log p(x|z) - \log p(z)] dz \tag{A.20}$$

$$\begin{aligned} &= -\int q(z|x) \left[\log \frac{q(z|x)}{p(z)} - \log p(x|z) \right] dz \\ &= \int q(z|x) \log p(x|z) dz - \int q(z|x) \log \frac{q(z|x)}{p(z)} dz \\ \text{ELBO} &= \underbrace{\mathbb{E}_{q(z|x)}[\log p(x|z)]}_{Reconstruction} - \underbrace{D_{KL}[q(z|x)||p(z)]}_{Complexity} \end{aligned} \tag{A.21}$$

the last decomposition provides an intuitive view of the ELBO through the trade-

off between its two terms. The reconstruction term assesses the capacity of the variational density $q(z|x)$ to generate latent samples z that can be transformed back into the original x. While the complexity acts as a penalty or cost, regularising the objective. It ensures that the variational density $q(z|x)$ does not deviate too far away from the prior $p(z)$.

It is also now possible to recontextualise equation A.18 in terms of the ELBO as

$$D_{KL}[q(z|x)||p(z|x)] = -\text{ELBO} + \log p(x) \tag{A.22}$$

Because a $D_{KL} \geq 0$ then

$$\begin{aligned} \log p(x) &= \text{ELBO} + D_{KL}[q(z|x)||p(z|x)] \\ \log p(x) &\geq \text{ELBO} \end{aligned} \tag{A.23}$$

which confirms the role of the ELBO in providing a lower bound for the marginal likelihood or *evidence* $p(x)$.

From a practical perspective, a variational autoencoder (Kingma and Welling, 2014; Rezende et al., 2014) is a way to implement the principles of variational inference through neural networks. The variational density is parameterised by the weights ϕ as $q_\phi(z|x)$. Then the process consists in training the networks to find the set of weights that maximises the ELBO

$$\phi^\star = \text{argmax}_\phi \text{ELBO} \tag{A.24}$$

A.3.1 Deriving the ELBO via the log marginal likelihood

In the inequality A.23 it was shown that the ELBO in addition to providing a tractable objective for approximating $p(z|x)$ also provides an approximation of the marginal itself. Being able to estimate the marginal is especially relevant, for instance, in unsupervised learning where a core idea is to perform density estimation on a dataset. By approximating the underlying data distribution $p(x)$ then it is possible to have a generative model from where to sample data. Besides the derivation from the variational objective in A.17, it is also possible to derive the ELBO directly from the log marginal in the following way

$$\begin{aligned}
\log p(x) &= \log \int p(x|z)p(z)dz \\
&= \log \int \frac{q(z|x)}{q(z|x)} p(x|z)p(z)dz \\
&= \log \int \frac{p(x|z)}{q(z|x)} q(z|x)p(z)dz \\
&= \log \mathbb{E}_{q(z|x)}\left[\frac{p(x|z)}{q(z|x)} p(z)\right] \\
&\geq \mathbb{E}_{q(z|x)}\left[\log \frac{p(x,z)}{q(z|x)}\right] \triangleq \text{ELBO}
\end{aligned} \tag{A.25}$$

B Bootstrapped Transition Functions and Error-Correction Architecture

The model of the environment is learned through a multi-layer Convolutional Neural Network (CNN) followed by a fully-connected layer. The CNN receives as input the current observation of size 15 x 19 x 3, and it is concatenated with a broadcasted one-hot representation of the last action selected by the policy. The CNN consists of an initial convolutional layer followed by two blocks, each formed by a *pool-and-inject* layer that attempts to capture long-range temporal dependencies (Racanière et al., 2017). This layer acts by applying a max-pool operation to the feature maps and broadcasts the result by preserving the size and concatenating it with the original input. The output from the pool-and-inject layer is passed through two double convolutional layers in parallel. Both outputs are concatenated and passed further to another convolutional layer. Then the output is concatenated with the original input and passed on to the next basic block. Finally, the output travels separately through another convolutional layer and a fully-connected layer on one hand, and on the other through two convolutional and a fully connected layer, to produce the next predicted observation and reward respectively. ReLU is used for all the non-linearities in the network.

The network output corresponds to the categorical representation of every cell in the next predicted frame and the predicted reward. The frame and the reward are transformed into their original format and passed on to the agent to continue the cycle. The network is trained to predict these outputs by stochastic gradient

descent on the cross-entropy loss between predictions and actual data.

Accordingly, BTF and BTF+RPF multi-headed networks generate K frames and rewards. As outlined in section 3.4.2, RPFs require a prior provided by an additional neural network with fixed parameters. The prior network maintains the same structure, inputs, and outputs as the bootstrapped network. The full list of hyperparameters used during model learning and planning is listed in table B.1.

Environment Model			
Training steps	50000	Number of heads	10
Mask sampling	0.5	Prior scale (RPF)	1
Optimiser	Adam	Learning rate	1e-5
Gradnorm	0.5	Buffer size	100000
Minibatch size	32	Min buffer size	5000
Rolling Horizon Evolution			
Sequence length	1, 5, 10, 20	Population size	10
Mutation rate	0.9	Shift-buffer	True

Table B.1: *Hyperparameters.*

C | Expected Free Energy with measurements v

We consider a generative model $p(s, o, v|\pi)$ for the EFE equation and obtain a joint distribution of preferences $p(o, v)$. If we are interested exclusively in v, assuming and treating o and v as if they were independent modalities, and ignoring o we obtain:

$$\begin{aligned} \mathrm{G}(\pi, \tau) &= \mathbb{E}_{q(o_\tau, v_\tau, \theta, s_\tau|\pi)}[\ln q(s_\tau|\pi) - \ln p(s_\tau, o_\tau, v_\tau|\pi)] \qquad \text{(C.1)} \\ &\approx \mathbb{E}_{q(o_\tau, v_\tau, \theta, s_\tau|\pi)}[\ln q(s_\tau|\pi) - \ln q(s_\tau|, o_\tau, v_\tau, \pi) - \ln p(o_\tau, v_\tau)] \\ &\approx \mathbb{E}_{q(o_\tau, v_\tau, \theta, s_\tau|\pi)}[\ln q(s_\tau|\pi) - \ln q(s_\tau|, o_\tau, v_\tau, \pi) - \ln p(o_\tau) - \ln p_\theta(v_\tau)] \\ &\approx \mathbb{E}_{q(o_\tau, v_\tau, \theta, s_\tau|\pi)}[\ln q(s_\tau|\pi) - \ln q(s_\tau|, o_\tau, v_\tau, \pi) - \ln p_\theta(v_\tau)] \\ &\approx -\mathbb{E}_{q(o_\tau, v_\tau, \theta|\pi)} D_{KL}[q(s_\tau|, o_\tau, v_\tau, \pi)||q(s_\tau|\pi)] - \mathbb{E}_{q(v_\tau, \theta, s_\tau|\pi)}[\ln p_\theta(v_\tau)] \\ & \qquad \text{(C.2)} \end{aligned}$$

D Novelty and salience

The derivation is equivalent to those found in the classical tabular descriptions of active inference where instead of learning transitions via a function approximator, a mapping from hidden states to observations is encoded by a likelihood matrix A. In the tabular case, the beliefs of the probability of an observation given a state are contained in the parameters a_{ij}, which are updated as the agent obtains a particular observation.

$$
\begin{aligned}
\mathrm{G}(\pi,\tau) &= \mathbb{E}_{q(o_\tau,s_\tau,v_\tau,\phi|\pi)}[\ln q(s_\tau,\phi|\pi) - \ln p(o_\tau,v_\tau,s_\tau,\phi|\pi)] \\
&= \mathbb{E}_{q(o_\tau,s_\tau,v_\tau,\phi|\pi)}[\ln q(\phi) + \ln q(s_\tau|\pi) \\
&\quad - \ln p(\phi|s_\tau,o_\tau,v_\tau,\pi) - \ln p(s_\tau|o_\tau,v_\tau,\pi) - \ln p(o_\tau,v_\tau)] \\
&\approx \mathbb{E}_{q(o_\tau,s_\tau,v_\tau,\phi|\pi)}[\ln q(\phi) + \ln q(s_\tau|\pi) \\
&\quad - \ln q(\phi|s_\tau,o_\tau,v_\tau,\pi) - \ln q(s_\tau|o_\tau,v_\tau,\pi) - \ln p_\theta(v_\tau)] \\
&\approx \mathbb{E}_{q(o_\tau,s_\tau,v_\tau,\phi|\pi)}[\ln q(s_\tau|\pi) - \ln q(s_\tau|o_\tau,v_\tau,\pi)] \\
&\quad + \mathbb{E}_{q(o_\tau,s_\tau,v_\tau\phi|\pi)}[\ln q(\phi) - \ln q(\phi|s_\tau,o_\tau,v_\tau,\pi)] \\
&\quad - \mathbb{E}_{q(o_\tau,s_\tau,v_\tau,\phi|\pi)}[\ln p(v_\tau)] \\
&\approx -\mathbb{E}_{q(o_\tau,s_\tau,v_\tau,\phi|\pi)}[\ln q(s_\tau|o_\tau,v_\tau,\pi) - \ln q(s_\tau|\pi)] \\
&\quad - \mathbb{E}_{q(o_\tau,s_\tau,v_\tau,\phi|\pi)}[\ln q(\phi|s_\tau,o_\tau,v_\tau,\pi) - \ln q(\phi)] \\
&\quad - \mathbb{E}_{q(o_\tau,s_\tau,v_\tau,\phi|\pi)}[\ln p(v_\tau)] \\
&\approx -\underbrace{\mathbb{E}_{q(o_\tau,v_\tau,\phi|\pi)}\left[D_{KL}[q(s_\tau|o_\tau,v_\tau,\pi)||q(s_\tau|\pi)]\right]}_{salience} \\
&\quad - \underbrace{\mathbb{E}_{q(o_\tau,v_\tau,s_\tau|\pi)}\left[D_{KL}[q(\phi|s_\tau,o_\tau,v_\tau,\pi)||q(\phi)]\right]}_{novelty} \\
&\quad - \underbrace{\mathbb{E}_{q(o_\tau,v_\tau,s_\tau,\phi|\pi)}[\ln p(v_\tau)]}_{instrumental\ value}
\end{aligned}
\tag{D.1}
$$

E Active Inference Implementation

We tested on the Flappy Bird environment (Tasfi, 2016). The environment sends a non-visual vector of features containing:

- the bird y position
- the bird velocity.
- next pipe distance to the bird
- next pipe top y position
- next pipe bottom y position
- distance from the bird to the pipe after the next
- top y position of the pipe after the next
- bottom y position of the pipe after the next

The parameter θ of the Bernoulli distribution $p(v)$ was estimated from a *measurement buffer* (i.e. queue) containing the last N values of v gathered by the agent. We tested the agents with large buffers (e.g. 20^6) as well as small buffers (e.g. 20) without significant change in performance. The results reported in fig. 7.4 and 7.5 were obtained with small-sized buffers as specified in the hyperparameter table below.

The DQN agent was trained to approximate with a neural network a Q-function $Q_\phi(\{s, \theta\}, .)$. For our case study $s = o$, and contains the vector of features. θ is the parameter corresponding to the currently estimated statistics of $p(v)$. An action is sampled uniformly with probability ϵ otherwise $a_t = \min_a Q_\phi(\{s_t, \theta_t\}, a)$. ϵ decays during training.

For the EFE agent, the transition model $p(s_t|s_{t-1}, \phi, \pi)$ is implemented as a $\mathcal{N}(\{s_t, \theta_t\}; f_\phi(s_{t-1}, \theta_{t-1}, a_{t-1}), f_\phi(s_{t-1}, \theta_{t-1}, a_{t-1}))$.

Where a is an action of a current policy π with one-hot encoding and f_ϕ is an ensemble of K neural networks which predicts the next values of s and θ. The surprisal model is also implemented with a neural network and trained to predict directly the surprisal in the future as $f_\xi(s_{t-1}, \theta_{t-1}, a_{t-1}) = -\ln p_{\theta_{t-1}}(v_t)$.

To calculate the expected free energy in equation 7.11 from a simulated sequence of future steps, we follow the approach described in appendix G in Tschantz et al. (2020) where they show that an information gain of the form $\mathbb{E}_{q(s|\phi)} D_{KL}[q(\phi|s)||q(\phi)]$ can be decomposed as,

$$\mathbb{E}_{q(s|\phi)} D_{KL}[q(\phi|s)||q(\phi)] = -\mathbb{E}_q(\phi) H[q(s|\phi)] + H[\mathbb{E}_{q(\phi)} q(s|\phi)] \tag{E.1}$$

with the first term computed analytically from the ensemble output and the second term approximated with a k-NN estimator (Beirlant et al., 1997).

Hyperparameters	**DQN**	**EFE**
Measurement v buffer size	20	20
Replay buffer size	10^6	10^6
Batch size	64	50
Learning rate	1^{-3}	1^{-3}
Discount rate	0.99	-
Final ϵ	0.01	-
Seed episodes	5	3
Ensemble size	-	25
Planning horizon	-	15
Number of candidates	-	500
Mutation rate	-	0.5
Shift buffer	-	True

F | Drive decomposition

$$
\begin{aligned}
\mathrm{G}(\pi, \tau) &= \mathbb{E}_{q(o_\tau, v_\tau, \theta, s_\tau|\pi)}[\ln q(s_\tau|\pi) - \ln p(s_\tau, o_\tau, v_\tau|\pi)] \\
&= \mathbb{E}_{q(o_\tau, v_\tau, \theta, s_\tau|\pi)}[\ln q(s_\tau|\pi) - \ln p(v_\tau|s_\tau, o_\tau, \pi) - \ln p(s_\tau, o_\tau|\pi)] \\
&= \mathbb{E}_{q(o_\tau, v_\tau, \theta, s_\tau|\pi)}[\ln q(s_\tau|\pi) - \ln p(s_\tau|o_\tau, \pi) - \ln p(o_\tau) - \ln p(v_\tau|s_\tau, o_\tau, \pi)] \\
&\approx \mathbb{E}_{q(o_\tau, v_\tau, \theta, s_\tau|\pi)}[\ln q(s_\tau|\pi) - \ln p(s_\tau|o_\tau, \pi)] - \mathbb{E}_{q(o_\tau, v_\tau, \theta, s_\tau|\pi)}[\ln p(o_\tau)] \\
&\quad + \mathbb{E}_{q(o_\tau, s_\tau|\pi)} H[p(v_\tau|s_\tau, o_\tau, \pi)]
\end{aligned}
\tag{F.1}
$$

G State-Space Models architectures

Observation embedding: A $3 \times 64 \times 64$ or a $3 \times 28 \times 28$ array, for the global and local case respectively, passes through a four-layer CNN with output sizes of 32, 64, 128 and 256, kernels of 4×4 and stride 2. The exception is when the network receives a local observation, if that occurs, the kernel in the last convolutional layer is of size 1. All the activation functions are ReLU. The output is passed further to a linear layer to produce an embedding of size 100.

Transition model: An embedding of the stochastic state, action, and deterministic state is passed through a GRU to obtain the next deterministic state. The next stochastic state is sampled by applying the reparameterisation trick.

Observation model: For decoding an observation an embedding of the latent states is passed through a deconvolutional network with four layers with outputs 128, 64, 32, and 3. The kernels 5, 5, 6, and 6, and stride 2 for global observations. For local observations, the kernels are 2, 3, 5, and 4. ReLU is used for all activation functions.

Reward model: The embedding of the latent states is passed through a three-layer feed-forward neural network with hidden size 100 and ReLU activation functions.

G.1 Q-Model SSMs

Q-value model: Follows the same construction as the reward model.

G.2 NR-SSMs

For the InfoNCE objective, we use a bilinear model with an output of size 100.

G.3 Hyperparameters

	SSM	**NR-SSM**
Training		
World Model learning rate	2e-4	2e-4
Value learning rate	8e-5	8e-5
NCE learning rate	-	3e-4
Epsilon	1e-4	1e-4
KL weight	1	1
KL balance	0.7	0.7
Embeddings		
Observation	100	100
Deterministic	100	100
Stochastic	32	32
Replay buffer		
Buffer size	10000	10000
Seed steps	4000	4000
Batch size	50	50
Batch sequence size	8	8
Planning		
Planning horizon	20	20
Candidates	300	300
Mutation rate	0.5	0.5
Shift buffer	True	True

H | InfoNCE

InfoNCE (van den Oord et al., 2019) belongs to a family of unsupervised loss functions known as Noise Contrastive Estimation (NCE) (Gutmann and Hyvärinen, 2010). Originally the objective of these methods was to approximate a conditional model $p(x|c)$, for instance, to be able to predict x given context c. These methods simplify and substitute the problem of estimating $p(x|c)$ with the problem of learning a binary classifier to determine whether a data point has been sampled from $p(x|c)$ or from an arbitrarily defined *noise* distribution $q(x)$. We form *positive* pairs (x, c) sampled from the $p(x|c)$ and assign them the label $Y = 1$. Correspondingly, we can also form *negative* pairs (x', c) where x' has been sampled from the noise distribution $q(x)$ and assign the label $Y = 0$. The NCE objective maximises the log-likelihood $p(Y|x, c)$ and learns a score function $f_\theta(x, c) \approx p(x|c)$[1] which associates larger scores to positive pairs and lower scores to negative pairs.

This general approach has been extended to a multi-class setting (Jozefowicz et al., 2016; Sohn, 2016). Instead of determining whether a single sample comes from the conditional or from the noise distribution, the question becomes which of the k samples comes from the conditional distribution $p(x|c)$. This leads to some important differences. In the loss function, the score function that arises does not approximate the conditional model $p(x|c)$ but instead, it is proportional

[1] This is assuming self-normalisation $Z_\theta(c) \approx 1$.

to a density ratio $\frac{p(x|c)}{p(x)}$. The loss function is defined as:

$$\mathcal{L} = -\mathbb{E}\left[\log \frac{f_\theta(x,c)}{\sum_{x' \in X} f_\theta(x',c)}\right] \tag{H.1}$$

Where the score function $f_\theta = exp(g_\theta(x,c))$ and $g_\theta()$ corresponds to a similarity function. InfoNCE assumes that the noise distribution $q(x)$ is the marginal $p(x)$, and relates the ratio $f_\theta(x,c) \propto \frac{p(x|c)}{p(x)}$ and the loss function to the information that we gain when we provide c to predict x. We can illustrate it as

$$I(x;c) = \sum_{x,c} p(x,c) \log \frac{p(x|c)}{p(x)} \propto \frac{p(x|c)}{p(x)} = f_\theta(x,c) \tag{H.2}$$

This connection implies that the minimisation of the loss function maximises a lower bound on the mutual information. Thus similar to NCE, the score function measure the degree to which x is associated with c. In practice, InfoNCE has been extensively used in representation learning to train neural networks to learn to produce embeddings that preserve information between positive pairs. This is also the context in which we have applied InfoNCE in this thesis, to preserve the part of the present and past that informs about the agent's future latent state.

8 Bibliography

Agostini, A. and Celaya, E. (2010). Reinforcement Learning with a Gaussian mixture model. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Aguilera, M., Millidge, B., Tschantz, A., and Buckley, C. L. (2022). How particular is the physics of the free energy principle? *Physics of Life Reviews*, 40:24–50.

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. (2016). Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*.

Anand, A., Racah, E., Ozair, S., Bengio, Y., Côté, M.-A., and Hjelm, R. D. (2020). Unsupervised State Representation Learning in Atari. *arXiv:1906.08226 [cs, stat]*.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. (2018). Hindsight Experience Replay.

Asadi, K., Misra, D., Kim, S., and Littman, M. L. (2019). Combating the Compounding-Error Problem with a Multi-step Model. *arXiv:1905.13320 [cs, stat]*.

Ashby, W. R. (1958). Requisite variety and its implications for the control of complex systems. In *Facets of Systems Science*, pages 405–417. Springer.

Attias, H. (2003). Planning by Probabilistic Inference. In *International Workshop on Artificial Intelligence and Statistics*, pages 9–16. PMLR.

Ay, N., Bertschinger, N., Der, R., Güttler, F., and Olbrich, E. (2008). Predictive information and explorative behavior of autonomous robots. *The European Physical Journal B*, 63(3):329–339.

Badia, A. P., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, D., and Blundell, C. (2020). Agent57: Outperforming the Atari Human Benchmark.

Bamford, C., Huang, S., and Lucas, S. (2020). Griddly: A platform for AI research in games.

Barandiaran, X. E., Paolo, E. D., and Rohde, M. (2009). Defining Agency: Individuality, Normativity, Asymmetry, and Spatio-temporality in Action:. *Adaptive Behavior*.

Barber, D. and Agakov, F. (2003). The IM algorithm: A variational approach to Information Maximization. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, pages 201–208, Cambridge, MA, USA. MIT Press.

Beal, M. J. (2003). *Variational Algorithms for Approximate Bayesian Inference*. Doctoral, UCL (University College London).

Beer, R. D. and Williams, P. L. (2015). Information Processing and Dynamics in Minimally Cognitive Agents. *Cognitive Science*, 39(1):1–38.

Beirlant, J., Dudewicz, E. J., Györfi, L., and Dénes, I. (1997). Nonparametric entropy estimation. An overview. *International Journal of Mathematical and Statistical Sciences*, 6(1):17–39.

Belghazi, M. I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, D. (2018). Mutual Information Neural Estimation. In *Proceedings of the 35th International Conference on Machine Learning*, pages 531–540. PMLR.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. (2016). Unifying count-based exploration and intrinsic motivation. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 1471–1479. Curran Associates, Inc.

Bellemare, M., Veness, J., and Bowling, M. (2013). Bayesian Learning of Recursively Factored Environments. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1211–1219. PMLR.

Bellemare, M., Veness, J., and Talvitie, E. (2014). Skip Context Tree Switching. In *Proceedings of the 31st International Conference on Machine Learning*, pages 1458–1466. PMLR.

Bellman, R. (1952). On the Theory of Dynamic Programming. *Proceedings of the National Academy of Sciences of the United States of America*, 38(8):716–719.

Bellman, R. (1957). A Markovian Decision Process. *Journal of Mathematics and Mechanics*, 6(5):679–684.

Berseth, G., Geng, D., Devin, C., Rhinehart, N., Finn, C., Jayaraman, D., and Levine, S. (2020). SMiRL: Surprise Minimizing RL in Dynamic Environments. *arXiv:1912.05510 [cs, stat]*.

Bertsekas, D. (1995). *Dynamic Programming and Optimal Control: Volume I*, volume 4. Athena scientific.

Bialek, W., De Ruyter Van Steveninck, R. R., and Tishby, N. (2006). Efficient representation as a design principle for neural coding and computation. In *2006 IEEE International Symposium on Information Theory*, pages 659–663.

Bialek, W., Nemenman, I., and Tishby, N. (2001). Complexity through nonextensivity. *Physica A: Statistical Mechanics and its Applications*, 302(1):89–99.

Bialek, W. and Tishby, N. (1999). Predictive Information.

Biehl, M., Guckelsberger, C., Salge, C., Smith, S. C., and Polani, D. (2018). Expanding the Active Inference Landscape: More Intrinsic Motivations in the Perception-Action Loop. *Frontiers in Neurorobotics*, 12:45.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 1613–1622, Lille, France. JMLR.org.

Botvinick, M. and Toussaint, M. (2012). Planning as inference. *Trends in Cognitive Sciences*, 16(10):485–488.

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

Buesing, L., Weber, T., Racaniere, S., Eslami, S. M. A., Rezende, D., Reichert, D. P., Viola, F., Besse, F., Gregor, K., Hassabis, D., and Wierstra, D. (2018). Learning and Querying Fast Generative Models for Reinforcement Learning.

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

Cao, F. J. and Feito, M. (2009). Thermodynamics of feedback controlled systems. *Physical Review E*, 79(4):041118.

Carver, C. S. and Scheier, M. F. (1981). Cybernetics, Information, and Control. In Carver, C. S. and Scheier, M. F., editors, *Attention and Self-Regulation: A Control-Theory Approach to Human Behavior*, SSSP Springer Series in Social Psychology, pages 11–31. Springer, New York, NY.

Çatal, O., Nauta, J., Verbelen, T., Simoens, P., and Dhoedt, B. (2019). Bayesian policy selection using active inference. *arXiv:1904.08149 [cs]*.

Çatal, O., Wauthier, S., Verbelen, T., De Boom, C., and Dhoedt, B. (2020). Deep Active Inference for Autonomous Robot Navigation.

Chen, C., Wu, Y.-F., Yoon, J., and Ahn, S. (2022). TransDreamer: Reinforcement Learning with Transformer World Models.

Chiappa, S., Racaniere, S., Wierstra, D., and Mohamed, S. (2017). Recurrent Environment Simulators.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014a). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014b). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv:1406.1078 [cs, stat]*.

Cisek, P. (2007). Cortical mechanisms of action selection: The affordance competition hypothesis. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 362(1485):1585–1599.

Cobbe, K., Klimov, O., Hesse, C., Kim, T., and Schulman, J. (2019). Quantifying Generalization in Reinforcement Learning.

Corbetta, M. and Shulman, G. L. (2002). Control of goal-directed and stimulus-driven attention in the brain. *Nature Reviews Neuroscience*, 3(3):201–215.

Cos, I., Cañamero, L., Hayes, G. M., and Gillies, A. (2013). Hedonic value: Enhancing adaptation for motivated agents. *Adaptive Behavior*, 21(6):465–483.

Coulom, R. (2007). Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In van den Herik, H. J., Ciancarini, P., and Donkers, H. H. L. M. J., editors, *Computers and Games*, Lecture Notes in Computer Science, pages 72–83, Berlin, Heidelberg. Springer.

Crutchfield, J. P. and Packard, N. H. (1983). Symbolic dynamics of noisy chaos. *Physica D: Nonlinear Phenomena*, 7(1):201–223.

Cuccu, G., Togelius, J., and Cudre-Mauroux, P. (2019). Playing Atari with Six Neurons.

Damasio, A. R. (2004). Emotions and Feelings: A Neurobiological Perspective. In *Feelings and Emotions: The Amsterdam Symposium*, Studies in Emotion and Social Interaction, pages 49–57. Cambridge University Press, New York, NY, US.

Dayan, P. (1993). Improving Generalization for Temporal Difference Learning: The Successor Representation. *Neural Computation*, 5(4):613–624.

Dayan, P. (2012). How to set the switches on this thing. *Current Opinion in Neurobiology*, 22(6):1068–1074.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. (1995). The Helmholtz machine. *Neural Computation*, 7(5):889–904.

Deisenroth, M. P. and Rasmussen, C. E. (2011). PILCO: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 465–472, Madison, WI, USA. Omnipress.

Di Paolo, E. A. (2003). Organismically-inspired robotics : Homeostatic adaptation and teleology beyond the closed sensorimotor loop.

Di Paolo, E. A. (2005). Autopoiesis, Adaptivity, Teleology, Agency. *Phenomenology and the Cognitive Sciences*, 4(4):429–452.

Di Paolo, E. A. (2010). Robotics Inspired in the Organism. *Intellectica. Revue de l'Association pour la Recherche Cognitive*, 53(1):129–162.

Donsker, M. D. and Varadhan, S. R. S. (1976). Asymptotic evaluation of certain Markov process expectations for large time—III. *Communications on Pure and Applied Mathematics*, 29(4):389–461.

Drivdahl, S. B. and Hyman, I. E. (2014). Fluidity in autobiographical memories: Relationship memories sampled on two occasions. *Memory*, 22(8):1070–1081.

Duane, S., Kennedy, A. D., Pendleton, B. J., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics Letters B*, 195(2):216–222.

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

Efron, B. (1979). Bootstrap Methods: Another Look at the Jackknife. *The Annals of Statistics*, 7(1):1–26.

Eysenbach, B., Zhang, T., Salakhutdinov, R., and Levine, S. (2022). Contrastive Learning as Goal-Conditioned Reinforcement Learning.

Fan, J. and Xiao, C. (2022). Generalized Data Distribution Iteration.

Fan, J., Xiao, C., and Huang, Y. (2022). GDI: Rethinking What Makes Reinforcement Learning Different From Supervised Learning.

Farebrother, J., Machado, M. C., and Bowling, M. (2020). Generalization and Regularization in DQN.

Farquhar, G., Rocktäschel, T., Igl, M., and Whiteson, S. (2018). TreeQN and ATreeC: Differentiable Tree-Structured Models for Deep Reinforcement Learning.

Friston, K. (2012). A Free Energy Principle for Biological Systems. *Entropy (Basel, Switzerland)*, 14(11):2100–2121.

Friston, K. (2013). Life as we know it. *Journal of The Royal Society Interface*, 10(86):20130475.

Friston, K., Adams, R., and Montague, R. (2012a). What is value—accumulated reward or evidence? *Frontiers in Neurorobotics*, 6.

Friston, K., FitzGerald, T., Rigoli, F., Schwartenbeck, P., and Pezzulo, G. (2016). Active Inference: A Process Theory. *Neural Computation*, 29(1):1–49.

Friston, K., Kilner, J., and Harrison, L. (2006). A free energy principle for the brain. *Journal of Physiology-Paris*, 100(1):70–87.

Friston, K., Samothrakis, S., and Montague, R. (2012b). Active inference and agency: Optimal control without cost functions. *Biological Cybernetics*, 106(8):523–541.

Friston, K. J., Daunizeau, J., and Kiebel, S. J. (2009). Reinforcement Learning or Active Inference? *PLoS ONE*, 4(7):e6421.

Friston, K. J., Lin, M., Frith, C. D., Pezzulo, G., Hobson, J. A., and Ondobaka, S. (2017). Active Inference, Curiosity and Insight. *Neural Computation*, 29(10):2633–2683.

Gaina, R. D., Lucas, S. M., and Perez-Liebana, D. (2017). Rolling horizon evolution enhancements in general video game playing. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 88–95.

Gal, Y. and Ghahramani, Z. (2016). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *International Conference on Machine Learning*, pages 1050–1059. PMLR.

Gao, S., Steeg, G. V., and Galstyan, A. (2015). Efficient Estimation of Mutual Information for Strongly Dependent Variables. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, pages 277–286. PMLR.

Garcia, C. E. and Morari, M. (1982). Internal model control. A unifying review and some new results. *Industrial & Engineering Chemistry Process Design and Development*, 21(2):308–323.

Geman, S. and Geman, D. (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741.

Ghahramani, Z. and Beal, M. J. (2000). Graphical Models and Variational Methods. In *Advanced Mean Field Methods - Theory and Practice*. MIT Press.

Ghugare, R., Bharadhwaj, H., Eysenbach, B., Levine, S., and Salakhutdinov, R. (2023). Simplifying Model-based RL: Learning Representations, Latent-space Models, and Policies with One Objective.

Gibson, J. J. (1979). *The Ecological Approach to Visual Perception*. The Ecological Approach to Visual Perception. Houghton, Mifflin and Company, Boston, MA, US.

Grassberger, P. (1986). Toward a quantitative theory of self-generated complexity. *International Journal of Theoretical Physics*, 25(9):907–938.

Grimm, C., Barreto, A., Farquhar, G., Silver, D., and Singh, S. (2021). Proper Value Equivalence.

Grimm, C., Barreto, A., Singh, S., and Silver, D. (2020). The value equivalence principle for model-based reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, pages 5541–5552, Red Hook, NY, USA. Curran Associates Inc.

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. (2016). Continuous Deep Q-Learning with Model-based Acceleration.

Guez, A., Mirza, M., Gregor, K., Kabra, R., Racanière, S., Weber, T., Raposo, D., Santoro, A., Orseau, L., Eccles, T., Wayne, G., Silver, D., and Lillicrap, T. (2019). An investigation of model-free planning. *arXiv:1901.03559 [cs, stat]*.

Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304. JMLR Workshop and Conference Proceedings.

Ha, D. and Schmidhuber, J. (2018). Recurrent World Models Facilitate Policy Evolution. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 2450–2462. Curran Associates, Inc.

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

Hafner, D., Lillicrap, T., Ba, J., and Norouzi, M. (2020). Dream to Control: Learning Behaviors by Latent Imagination.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. (2019). Learning Latent Dynamics for Planning from Pixels. In *International Conference on Machine Learning*, pages 2555–2565.

Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J. (2022). Mastering Atari with Discrete World Models.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A Formal Basis for the

Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107.

Henaff, M., Raileanu, R., Jiang, M., and Rocktäschel, T. (2022). Exploration via Elliptical Episodic Bonuses.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2019). Deep Reinforcement Learning that Matters.

Herreros, I. and Verschure, P. F. (2015). About the goal of a goals' goal theory. *Cognitive Neuroscience*, 6(4):218–219.

Hessel, M., Modayil, J., van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. (2017). Rainbow: Combining Improvements in Deep Reinforcement Learning.

Hinton, G. E. and van Camp, D. (1993). Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the Sixth Annual Conference on Computational Learning Theory*, COLT '93, pages 5–13, New York, NY, USA. Association for Computing Machinery.

Hinton, G. E. and Zemel, R. S. (1993). Autoencoders, minimum description length and Helmholtz free energy. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NIPS'93, pages 3–10, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. (2019). Learning deep representations by mutual information estimation and maximization.

Holland, G. Z., Talvitie, E. J., and Bowling, M. (2019). The Effect of Planning Shape on Dyna-style Planning in High-dimensional State Spaces.

Houthooft, R., Chen, X., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. (2016). VIME: Variational Information Maximizing Exploration. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, Mass.

Jaderberg, M., Mnih, V., Czarnecki, W. M., Schaul, T., Leibo, J. Z., Silver, D., and Kavukcuoglu, K. (2016). Reinforcement Learning with Unsupervised Auxiliary Tasks.

James, R. N. (2015). Exploration-exploitation: A cognitive dilemma still unresolved. *Cognitive Neuroscience*, 6(4):219–221.

Janner, M., Mordatch, I., and Levine, S. (2021). Generative Temporal Difference Learning for Infinite-Horizon Prediction.

Jordan, M. I., Ghahramani, Z., Jaakkola, T. S., and Saul, L. K. (1999). An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233.

Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. (2016). Exploring the Limits of Language Modeling.

Juechems, K. and Summerfield, C. (2019). Where Does Value Come From? *Trends in Cognitive Sciences*, 23(10):836–850.

Kaiser, L., Babaeizadeh, M., Milos, P., Osinski, B., Campbell, R. H., Czechowski, K., Erhan, D., Finn, C., Kozakowski, P., Levine, S., Mohiuddin, A., Sepassi, R., Tucker, G., and Michalewski, H. (2020). Model-Based Reinforcement Learning for Atari.

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

Kalman, R. E. and Bucy, R. S. (1961). New Results in Linear Filtering and Prediction Theory. *Journal of Basic Engineering*, 83(1):95–108.

Kappen, H. J., Gómez, V., and Opper, M. (2012). Optimal control as a graphical model inference problem. *Machine Learning*, 87(2):159–182.

Kendall, A. and Gal, Y. (2017). What uncertainties do we need in Bayesian deep learning for computer vision? In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 5580–5590, Long Beach, California, USA. Curran Associates Inc.

Keramati, M. and Gutkin, B. (2011). A Reinforcement Learning Theory for Homeostatic Regulation. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc.

Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Klyubin, A., Polani, D., and Nehaniv, C. (2005). Empowerment: A universal agent-centric measure of control. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 128–135 Vol.1.

Kolchinsky, A. and Wolpert, D. H. (2018). Semantic information, autonomous agency, and nonequilibrium statistical physics. *Interface Focus*, 8(6):20180041.

Konda, V. R. and Tsitsiklis, J. N. (1999). Actor-citic agorithms. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 1008–1014, Cambridge, MA, USA. MIT Press.

Kraskov, A., Stögbauer, H., and Grassberger, P. (2004). Estimating mutual information. *Physical Review E*, 69(6):066138.

Kulkarni, T., Gupta, A., Ionescu, C., Borgeaud, S., Reynolds, M., Zisserman, A., and Mnih, V. (2019). Unsupervised Learning of Object Keypoints for Perception and Control. *arXiv:1906.11883 [cs]*.

Kumar, N. M. (2018). Empowerment-driven Exploration using Mutual Information Estimation.

Laughlin, S. B., de Ruyter van Steveninck, R. R., and Anderson, J. C. (1998). The metabolic cost of neural information. *Nature Neuroscience*, 1(1):36–41.

Laurençon, H., Ségerie, C.-R., Lussange, J., and Gutkin, B. S. (2021). Continuous Homeostatic Reinforcement Learning for Self-Regulated Autonomous Agents.

Lee, A. X., Nagabandi, A., Abbeel, P., and Levine, S. (2020). Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33:741–752.

Leibfried, F., Kushman, N., and Hofmann, K. (2017). A Deep Learning Approach for Joint Video Frame and Reward Prediction in Atari Games.

Li, C., Chen, C., Carlson, D., and Carin, L. (2016). Preconditioned Stochastic Gradient Langevin Dynamics for deep neural networks. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, pages 1788–1794, Phoenix, Arizona. AAAI Press.

Li, W. and Todorov, E. (2004). Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems. In *First International Conference on Informatics in Control, Automation and Robotics*, pages 222–229.

Little, D. and Sommer, F. (2013). Learning and exploration in action-perception loops. *Frontiers in Neural Circuits*, 7.

Little, D. Y. and Sommer, F. T. (2011). Learning in embodied action-perception loops through exploration.

Lloyd, S. (1989). Use of mutual information to decrease entropy: Implications for the second law of thermodynamics. *Physical Review A*, 39(10):5378–5386.

Man, K. and Damasio, A. (2019). Homeostatically Motivated Intelligence for Feeling Machines. In *AAAI Spring Symposium: Towards Conscious AI Systems*.

Martius, G., Der, R., and Ay, N. (2013). Information Driven Self-Organization of Complex Robotic Behaviors. *PLOS ONE*, 8(5):e63400.

Martius, G., Jahn, L., Hauser, H., and Hafner, V. V. (2014). Self-exploration of the Stumpy Robot with Predictive Information Maximization. In del Pobil, A. P., Chinellato, E., Martinez-Martin, E., Hallam, J., Cervera, E., and Morales, A., editors, *From Animals to Animats 13*, Lecture Notes in Computer Science, pages 32–42, Cham. Springer International Publishing.

Martius, G. and Olbrich, E. (2015). Quantifying Emergent Behavior of Autonomous Robots. *Entropy*, 17(10):7266–7297.

Maturana, H. R. and Varela, F. J. (1987). *The Tree of Knowledge: The Biological Roots of Human Understanding*. The Tree of Knowledge: The Biological Roots of Human Understanding. New Science Library/Shambhala Publications, Boston, MA, US.

Maxwell, J. C. (1871). *Theory of Heat*. Cambridge Library Collection - Physical Sciences. Cambridge University Press, Cambridge.

McAllester, D. and Stratos, K. (2020a). Formal Limitations on the Measurement of Mutual Information. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, pages 875–884. PMLR.

McAllester, D. and Stratos, K. (2020b). Formal Limitations on the Measurement of Mutual Information. *arXiv:1811.04251 [cs, math, stat]*.

Menon, V. and Uddin, L. Q. (2010). Saliency, switching, attention and control: A network model of insula function. *Brain structure & function*, 214(5-6):655–667.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092.

Micheli, V., Alonso, E., and Fleuret, F. (2022). Transformers are Sample Efficient World Models.

Millidge, B. (2019). Combining Active Inference and Hierarchical Predictive Coding: A Tutorial Introduction and Case Study.

Millidge, B. (2020). Deep active inference as variational policy gradients. *Journal of Mathematical Psychology*, 96:102348.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing Atari with Deep Reinforcement Learning. *arXiv:1312.5602 [cs]*.

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

Mobin, S. A., Arnemann, J. A., and Sommer, F. (2014). Information-based learning by agents in unbounded state spaces. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Moerland, T. M., Broekens, J., Plaat, A., and Jonker, C. M. (2022). Model-based Reinforcement Learning: A Survey.

Mohamed, S. and Rezende, D. J. (2015). Variational information maximisation for intrinsically motivated reinforcement learning. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2125–2133, Cambridge, MA, USA. MIT Press.

Nash, R. A., Wade, K. A., Garry, M., and Adelman, J. S. (2017). A robust preference for cheap-and-easy strategies over reliable strategies when verifying personal memories. *Memory*, 25(7):890–899.

Nguyen, X., Wainwright, M. J., and Jordan, M. I. (2010). Estimating Divergence Functionals and the Likelihood Ratio by Convex Risk Minimization. *IEEE Transactions on Information Theory*, 56(11):5847–5861.

Niven, J. E. (2014). Neural Energetics: Hungry Flies Turn Down the Visual Gain. *Current Biology*, 24(8):R313–R315.

Oh, J., Guo, X., Lee, H., Lewis, R., and Singh, S. (2015). Action-conditional video prediction using deep networks in Atari games. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pages 2863–2871, Cambridge, MA, USA. MIT Press.

Oh, J., Singh, S., and Lee, H. (2017). Value Prediction Network. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Osband, I., Aslanides, J., and Cassirer, A. (2018). Randomized Prior Functions for Deep Reinforcement Learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 8617–8629. Curran Associates, Inc.

Ostrovski, G., Bellemare, M. G., Oord, A., and Munos, R. (2017). Count-Based Exploration with Neural Density Models. In *International Conference on Machine Learning*, pages 2721–2730. PMLR.

Oudeyer, P.-Y. and Kaplan, F. (2007). What is Intrinsic Motivation? A Typology of Computational Approaches. *Frontiers in Neurorobotics*, 1:6.

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. (2007). Intrinsic Motivation Systems for Autonomous Mental Development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–286.

Paolo, E. D., Thompson, E., and Beer, R. (2022). Laying down a forking path: Tensions between enaction and the free energy principle. *Philosophy and the Mind Sciences*, 3.

Parrondo, J. M. R., Horowitz, J. M., and Sagawa, T. (2015). Thermodynamics of information. *Nature Physics*, 11(2):131–139.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. (2017). Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pages 2778–2787, Sydney, NSW, Australia. JMLR.org.

Pathak, D., Gandhi, D., and Gupta, A. (2019). Self-Supervised Exploration via Disagreement. In *International Conference on Machine Learning*, pages 5062–5071. PMLR.

Pedraza, J. M., Garcia, D. A., and Pérez-Ortiz, M. F. (2018). Noise, Information and Fitness in Changing Environments. *Frontiers in Physics*, 6.

Perez-Liebana, D., Samothrakis, S., Lucas, S., and Rohlfshagen, P. (2013). Rolling horizon evolution versus tree search for navigation in single-player real-time games. In *GECCO '13*, pages 351–358.

Pezzulo, G. (2008). Coordinating with the Future: The Anticipatory Nature of Representation. *Minds and Machines*, 18(2):179–225.

Polani, D. (2009). Information: Currency of life? *HFSP journal*, 3(5):307–316.

Polani, D., Martinetz, T., and Kim, J. (2001). An Information-Theoretic Approach for the Quantification of Relevance. In Kelemen, J. and Sosík, P., editors, *Advances in Artificial Life*, Lecture Notes in Computer Science, pages 704–713, Berlin, Heidelberg. Springer.

Polani, D., Nehaniv, C. L., Martinetz, T., and Kim, J. T. (2006). Relevant information in optimized persistence vs. progeny strategies. In *Artificial Life X: Proceedings of The 10th International Conference on the Simulation and Synthesis of Living Systems, Bloomington IN*.

Poole, B., Ozair, S., Van Den Oord, A., Alemi, A., and Tucker, G. (2019a). On variational bounds of mutual information. In *International Conference on Machine Learning*, pages 5171–5180. PMLR.

Poole, B., Ozair, S., van den Oord, A., Alemi, A. A., and Tucker, G. (2019b). On Variational Bounds of Mutual Information. *arXiv:1905.06922 [cs, stat]*.

Racanière, S., Weber, T., Reichert, D., Buesing, L., Guez, A., Jimenez Rezende, D., Puigdomènech Badia, A., Vinyals, O., Heess, N., Li, Y., Pascanu, R., Battaglia, P., Hassabis, D., Silver, D., and Wierstra, D. (2017). Imagination-Augmented Agents for Deep Reinforcement Learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5690–5701. Curran Associates, Inc.

Ramstead, M. J. D., Badcock, P. B., and Friston, K. J. (2018). Answering Schrödinger's question: A free-energy formulation. *Physics of Life Reviews*, 24:1–16.

Ramstead, M. J. D., Constant, A., Badcock, P. B., and Friston, K. J. (2019). Variational ecology and the physics of sentient systems. *Physics of Life Reviews*, 31:188–205.

Rawlik, K., Toussaint, M., and Vijayakumar, S. (2010). Approximate Inference and Stochastic Optimal Control.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *International Conference on Machine Learning*, pages 1278–1286. PMLR.

Richalet, J., Rault, A., Testud, J. L., and Papon, J. (1978). Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14(5):413–428.

Ritter, H., Botev, A., and Barber, D. (2018). A Scalable Laplace Approximation for Neural Networks. https://iclr.cc/Conferences/2018/Schedule?showEvent=224.

Rubin, J., Shamir, O., and Tishby, N. (2012). Trading Value and Information in MDPs. In Guy, T. V., Kárný, M., and Wolpert, D. H., editors, *Decision Making with Imperfect Decision Makers*, Intelligent Systems Reference Library, pages 57–74. Springer, Berlin, Heidelberg.

Rubinstein, R. (1999). The Cross-Entropy Method for Combinatorial and Continuous Optimization. *Methodology And Computing In Applied Probability*, 1(2):127–190.

Rummery, G. A. and Niranjan, M. (1994). On-Line Q-Learning Using Connectionist Systems. Technical report.

Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice Hall, Upper Saddle River, NJ, 2nd edition edition.

Ryan, R. M. and Deci, E. L. (2000). Intrinsic and Extrinsic Motivations: Classic Definitions and New Directions. *Contemporary Educational Psychology*, 25(1):54–67.

Sagawa, T. and Ueda, M. (2012). Nonequilibrium thermodynamics of feedback control. *Physical Review E*, 85(2):021104.

Sajid, N., Ball, P. J., and Friston, K. J. (2020). Active inference: Demystified and compared. *arXiv:1909.10863 [cs, q-bio]*.

Salge, C., Glackin, C., and Polani, D. (2014). Empowerment–An Introduction. In Prokopenko, M., editor, *Guided Self-Organization: Inception*, Emergence, Complexity and Computation, pages 67–114. Springer, Berlin, Heidelberg.

Salge, C. and Guckelsberger, C. (2016). Does Empowerment Maximisation Allow for Enactive Artificial Agents? In *ALIFE 2016, the Fifteenth International Conference on the Synthesis and Simulation of Living Systems*, pages 704–711. MIT Press.

Sancaktar, C., van Gerven, M., and Lanillos, P. (2020). End-to-End Pixel-Based Deep Active Inference for Body Perception and Action. *arXiv:2001.05847 [cs, q-bio]*.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. (2015). Universal Value Function Approximators. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1312–1320. PMLR.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., Lillicrap, T., and Silver, D. (2020). Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model. *arXiv:1911.08265 [cs, stat]*.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal Policy Optimization Algorithms.

Shalizi, C. R. and Crutchfield, J. P. (2001). Computational Mechanics: Pattern and Prediction, Structure and Simplicity. *Journal of Statistical Physics*, 104(3):817–879.

Shridhar, K., Laumann, F., and Liwicki, M. (2019). A Comprehensive

guide to Bayesian Convolutional Neural Network with Variational Inference. *arXiv:1901.02731 [cs, stat]*.

Silver, D., Hasselt, H., Hessel, M., Schaul, T., Guez, A., Harley, T., Dulac-Arnold, G., Reichert, D., Rabinowitz, N., Barreto, A., and Degris, T. (2017a). The Predictron: End-To-End Learning and Planning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3191–3199. PMLR.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2017b). Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm.

Sohn, K. (2016). Improved Deep Metric Learning with Multi-class N-pair Loss Objective. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Song, J. and Ermon, S. (2020). Understanding the Limitations of Variational Mutual Information Estimators. *arXiv:1910.06222 [cs, math, stat]*.

Still, S. (2009). Information-theoretic approach to interactive learning. *EPL (Europhysics Letters)*, 85(2):28005.

Still, S. (2014). Information Bottleneck Approach to Predictive Inference. *Entropy*, 16(2):968–989.

Still, S., Sivak, D. A., Bell, A. J., and Crooks, G. E. (2012). Thermodynamics of Prediction. *Physical Review Letters*, 109(12):120604.

Stooke, A., Lee, K., Abbeel, P., and Laskin, M. (2021). Decoupling Representation Learning from Reinforcement Learning. In *Proceedings of the 38th International Conference on Machine Learning*, pages 9870–9879. PMLR.

Sutton, R. S. (1990). Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. In Porter, B. and Mooney, R., editors, *Machine Learning Proceedings 1990*, pages 216–224. Morgan Kaufmann, San Francisco (CA).

Sutton, R. S. (1991). Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163.

Sutton, R. S. and Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning, Second Edition: An Introduction*. MIT Press.

Szilard, L. (1929). über die Entropieverminderung in einem thermodynamischen System bei Eingriffen intelligenter Wesen. *Zeitschrift fur Physik*, 53:840–856.

Talvitie, E. (2014). Model regularization for stable sample rollouts. In *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence*, UAI'14, pages 780–789, Quebec City, Quebec, Canada. AUAI Press.

Talvitie, E. (2016). Self-Correcting Models for Model-Based Reinforcement Learning. https://arxiv.org/abs/1612.06018v2.

Tamar, A., WU, YI., Thomas, G., Levine, S., and Abbeel, P. (2016). Value Iteration Networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Tasfi, N. (2016). PyGame Learning Environment.

Taylor, M. E. and Stone, P. (2009). Transfer Learning for Reinforcement Learning Domains: A Survey. *The Journal of Machine Learning Research*, 10:1633–1685.

Taylor, S. F., Tishby, N., and Bialek, W. (2007). Information and fitness.

Tishby, N., Pereira, F. C., and Bialek, W. (2000). The information bottleneck method.

Tishby, N. and Polani, D. (2011). Information Theory of Decisions and Actions. In Cutsuridis, V., Hussain, A., and Taylor, J. G., editors, *Perception-Action Cycle: Models, Architectures, and Hardware*, Springer Series in Cognitive and Neural Systems, pages 601–636. Springer, New York, NY.

Todorov, E. (2008). General duality between optimal control and estimation. In *2008 47th IEEE Conference on Decision and Control*, pages 4286–4292.

Touchette, H. and Lloyd, S. (2000). Information-Theoretic Limits of Control. *Physical Review Letters*, 84(6):1156–1159.

Touchette, H. and Lloyd, S. (2004). Information-theoretic approach to the study of control systems. *Physica A: Statistical Mechanics and its Applications*, 331(1):140–172.

Tschantz, A., Baltieri, M., Seth, A. K., and Buckley, C. L. (2019). Scaling active inference. *arXiv:1911.10601 [cs, eess, math, stat]*.

Tschantz, A., Millidge, B., Seth, A. K., and Buckley, C. L. (2020). Reinforcement Learning through Active Inference. *arXiv:2002.12636 [cs, eess, math, stat]*.

Ueltzhöffer, K. (2018). Deep active inference. *Biological Cybernetics*, 112(6):547–573.

van den Oord, A., Li, Y., and Vinyals, O. (2019). Representation Learning with Contrastive Predictive Coding.

van der Waa, J., van Diggelen, J., van den Bosch, K., and Neerincx, M. (2018). Contrastive Explanations for Reinforcement Learning in terms of Expected Consequences.

van Seijen, H., Nekoei, H., Racah, E., and Chandar, S. (2020). The LoCA Regret: A Consistent Metric to Evaluate Model-Based Behavior in Reinforcement Learning. https://arxiv.org/abs/2007.03158v2.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Vezhnevets, A. S., Mnih, V., Agapiou, J., Osindero, S., Graves, A., Vinyals, O., and Kavukcuoglu, K. (2016). Strategic attentive writer for learning macro-actions. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 3494–3502, Red Hook, NY, USA. Curran Associates Inc.

von Uexküll, J. (1926). *Theoretical Biology*. Theoretical Biology. Harcourt, Brace & Co., Oxford, England.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards.* PhD thesis, University of Cambridge.

Watkins, C. J. C. H. and Dayan, P. (1992). Q-learning. *Machine Learning*, 8(3):279–292.

Watters, N., Matthey, L., Bosnjak, M., Burgess, C. P., and Lerchner, A. (2019). COBRA: Data-Efficient Model-Based RL through Unsupervised Object Discovery and Curiosity-Driven Exploration. *arXiv:1905.09275 [cs]*.

Weber, A. and Varela, F. J. (2002). Life after Kant: Natural purposes and the autopoietic foundations of biological individuality. *Phenomenology and the Cognitive Sciences*, 1(2):97–125.

Weber, T., Racanière, S., Reichert, D. P., Buesing, L., Guez, A., Rezende, D. J., Badia, A. P., Vinyals, O., Heess, N., Li, Y., Pascanu, R., Battaglia, P., Hassabis,

D., Silver, D., and Wierstra, D. (2018). Imagination-Augmented Agents for Deep Reinforcement Learning.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*.

Wells, G. L., Memon, A., and Penrod, S. D. (2006). Eyewitness Evidence: Improving Its Probative Value. *Psychological Science in the Public Interest*, 7(2):45–75.

Wiener, N. (1948). *Cybernetics; or Control and Communication in the Animal and the Machine*. Cybernetics; or Control and Communication in the Animal and the Machine. John Wiley, Oxford, England.

Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256.

Witten, I. H. (1977). An adaptive optimal controller for discrete-time Markov environments. *Information and Control*, 34(4):286–295.

Yoon, J., Wu, Y.-F., Bae, H., and Ahn, S. (2023). An Investigation into Pre-Training Object-Centric Representations for Reinforcement Learning.

Zhu, Z., Lin, K., Jain, A. K., and Zhou, J. (2023). Transfer Learning in Deep Reinforcement Learning: A Survey.