

Active Inference with Episodic Memory in the Animal-AI Environment

— Individual Project —

Alexey Zakharov
az519@ic.ac.uk

Supervisors: Dr. Matthew Crosby, Dr. Zafeirios Fountas

Imperial College London
Department of Computing

September 2020

Submitted in partial fulfillment of the requirements for the MSc degree in Artificial
Intelligence of Imperial College London

Abstract

Active inference is an emerging framework within computational neuroscience, which attempts to unify perception and action under the single objective of minimising the free-energy functional. Similar to model-based reinforcement learning, an active inference agent relies almost entirely on the characteristics and the quality of its internal model of the environment to make decisions. Specifically, to perform planning it simulates the possible future observations given a policy and chooses the path that produced the lowest values of a pre-defined objective function. Thus, the agent's decisions depend on how *accurate* the simulated observations are, as well as on how *informative* they are about the short- and long-term futures. However, both of these components pose major problems. First, learning a perfectly accurate generative model is not feasible in a sufficiently complex environment, which means that, ideally, an agent should train only on the most important features of the testbed. But how would an agent decide what to train on? Second, current state-of-the-art model-based methods suffer from sequential error accumulation and inefficient long-term predictions. This significantly limits their ability to predict farther into the future. Moreover, these methods do not offer the flexibility to *vary the timescale* over which the future is predicted. However, changing how far into the future an agent predicts depending on the context can be essential for an active inference agent to make more informed decisions. In this project, we address both of the outlined issues with the use of *episodic memory*. Empirical evidence from neuroscience suggests that episodic memory may carry a number of functional purposes, such as being necessary for the formation of new semantic memories, assisting humans in the imagination of future scenarios or keeping track of time. To address the first issue, we propose integration of the Prioritised Replay Buffer (PRB) into the framework of active inference. Originally applied to improve the speed of convergence of a model-free reinforcement learning system, we use prioritisation to incentivise the learning of a more action-oriented generative model. To address the second issue, we introduce the Temporal Episodic Memory (TEM) module, in which episodic memories are used to define a *subjective* timescale by which an agent perceives the environment. Our results demonstrate considerable improvement in the agent's performance on a number of tasks in the Animal-AI environment. PRB allowed the agent to learn better object-centric representations, while TEM resulted in a transition model that can systematically vary the temporal extent of its predictions, and is additionally able to imagine novel future scenarios.

Acknowledgements

I would like to thank my supervisors, Dr. Matthew Crosby and Dr. Zafeirios Fountas, who were the best supervisors I could have ever asked for. They supported me throughout this project by providing invaluable feedback on my ideas and by always being open-minded. Special thanks to my family who have been supportive of me throughout my studies and always motivated me to move forward.

Contents

1	Introduction	6
2	Background and Related Work	8
2.1	Animal-AI Environment	8
2.2	Model-based Reinforcement Learning	9
2.2.1	Preliminaries	9
2.2.2	Model-free vs. Model-based	10
2.2.3	Review of Model-based RL	11
2.2.4	Summary and Problem Identification	15
2.3	Active Inference	17
2.3.1	Preliminaries	17
2.3.2	Review of Active Inference Literature	20
2.4	Episodic Memory	26
2.4.1	Background and Motivation	26
2.4.2	Review of Episodic Memory in RL Literature	26
2.5	Summarising Remarks	30
2.6	Note on Legal and Professional Considerations	31
3	Baseline Architecture	32
4	Prioritised Replay Buffer	35
4.1	Details	35
4.2	Priority Metric	37
4.3	Experimental Results	40
5	Temporal Episodic Memory Module	45
5.1	Motivating Examples	45
5.2	Architecture	49
5.3	Memory Accumulation Inspection	53
5.4	TEM Imagination Roll-out Inspection	55
5.5	Experimental Results	60
6	Evaluation and Future Work	66
6.1	Contributions	66
6.2	Limitations and Future Work	66
7	Conclusion	70
A	Implementation of Prioritised Replay Buffer	81
B	Additional results of On-line vs. PRB comparison	83

C Architectural Details	84
C.1 On-line and PRB Agent	84
C.2 TEM Architecture	86
D PRB Transition Model Imagination	87

1 Introduction

Originating from the non-equilibrium steady-state physics, *the free-energy principle* is an elegant mathematical formulation, which attempts to provide a unified view of how the brain operates [1, 2]. In turn, this principle gave rise to the framework of *Active Inference* [3], which is being actively studied by researchers bordering neuroscience, cognitive science, and more recently artificial intelligence. Active inference has an ambitious goal of unifying perception and action under the single objective of minimising the free-energy functional. It has been demonstrated that this framework has the potential to tackle such important issues like exploration-exploitation trade-off or reward engineering [4], solutions for which naturally arise from the mathematical formulation of the free energy. Nevertheless, the interest for active inference has been largely limited to the neuroscience community, and the typical implementations overwhelmingly involve environments with discrete and small state-spaces. As a result, some attention has recently been directed to scaling the framework, and a few works have proposed its integration with deep learning methods.

Active inference shares some basic operational principles with model-based reinforcement learning (RL), in which an agent is similarly equipped with a model of its environment. At the same time, the field of model-based RL has seen great progress in the application of deep learning for parametrising an agent’s generative model [5]; however, little work has been done to apply these developments to active inference. As part of this project, we review the recent successes of the model-based RL and identify techniques with a strong potential for application in active inference. In Section 5, we will propose a novel way to learn a model of the environment with the use of autoregressive methods, which have been demonstrated to be effective in model-based RL [6].

At the same time, there is overwhelming evidence that episodic memory plays a crucial role in human learning and decision-making [7]. Although its specific function is still heavily debated, the majority of studies relate it to the human ability to imagine novel futures and perform mental time-travel [7]. Furthermore, episodic memory has been found to interplay with reward-based learning in humans [8], as well as to be necessary for the formation of new semantic memories [9]. The evidence for the breadth of functional purposes that episodic memory may capture makes it one of the most intriguing biological features of intelligence, and provides exciting avenues for research in the field of artificial intelligence. In this project, we investigate incorporation of episodic memory into an active inference agent and study its performance in the Animal-AI Environment. Specifically, we introduce and implement two separate techniques based on episodic memory, and empirically demonstrate the positive effects on the performance of an active inference agent.

This paper is structured so as to assist the reader in relating the different themes of the project. In Section 2 we review the literature on model-based reinforcement learning, active inference, and episodic memory. Additionally, we introduce the mathematical formalisms, where necessary. In Section 3 we describe the inner-workings of the baseline active inference agent from Fountas et al. [10] used throughout this project. In Section 4 we introduce our implementation of a prioritised replay buffer applied to active inference to improve the

quality of the agent’s generative model. Next, in Section 5 we present a novel approach to learning a transition dynamics model with the use of episodic memories. Finally, in Section 6 we provide a critical evaluation of the proposed systems and summarise our contributions.

2 Background and Related Work

This project touches on three main topics – model-based reinforcement learning, active inference or more broadly the free-energy principle, and episodic memory. We start by introducing the Animal-AI testbed in Section 2.1. In Section 2.2 we outline the general framework of model-based RL and cover the current state of the art. Next, in Section 2.3 we set the scene for active inference by discussing some of the latest work that has been done at the intersection with deep learning. In Section 2.4 we review the work that has been done to integrate episodic memory into RL agents and, more generally, into learning systems. Lastly, in Section 2.5 we summarise our findings from the review and formulate coherent conclusions with reference to the goals of this project.

2.1 Animal-AI Environment

The Animal-AI environment is a virtual testbed that can be used to assess cognitive capabilities of reinforcement learning agents (e.g. object permanence) [11, 12]. In particular, it was designed to test agents on out-of-training-distribution tasks and was largely inspired by the methodologies from comparative psychology and the studies of animal cognition. In this environment, an agent is tasked with reaching a reward, a green sphere, given a particular setup that may include obstacles (e.g. walls, movable objects), intermediary rewards (yellow spheres), penalising surfaces, translucent objects, etc. The important characteristic of this testbed, as the authors point out, is that an agent can be tested for the same cognitive ability in a variety of setups. This largely constraints the performance of over-fitting agents, encouraging them to learn more robust skills necessary for solving the tasks. This is in contrast to the majority of current benchmarks and test environments in reinforcement learning, which tend to assess for special-purpose skills instead [13].

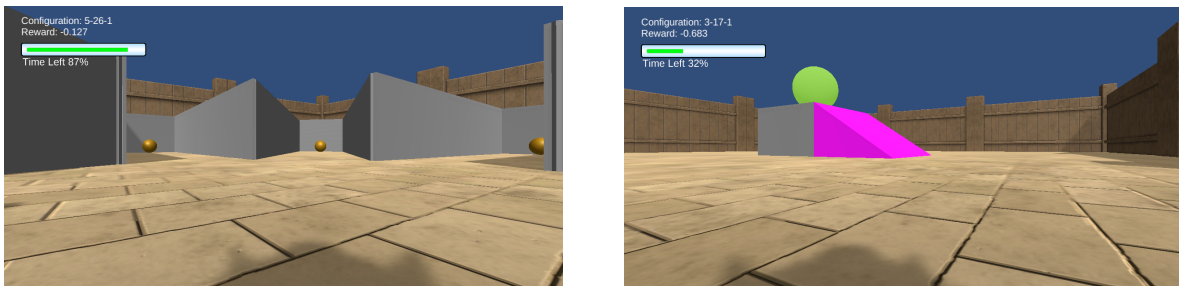


Figure 1: Examples of the Animal-AI environment setups [11].

One of the promising directions for solving out-of-distribution tasks in testbeds such as the Animal-AI is if an agent possesses an internal model of the environment and is able to plan its actions with respect to this model [14]. This way, an agent would be able to devise its next move depending on the context it finds itself in and the learned model of the environment’s dynamics (i.e. how any given action would change the state of the environment given the current state). Thus, model-based agents are often cited as being more capable of generalisation due to their ability to predict consequences of their actions, which results in them being more robust to previously unseen situations.

2.2 Model-based Reinforcement Learning

To contextualise the following review of model-based reinforcement learning, it is pertinent to reiterate its main connection to the framework of active inference. Active inference (Section 2.3) relies almost entirely on an agent’s generative model defined over a partially-observable Markov decision process (POMDPs), which is used to imagine possible future observations. In turn, Markov decision processes (MDPs) form the foundation of model-based RL research, with transition (forward) models being its primary focus. Thus, active inference can borrow from the rich body of research that is being done in this field such as learning more accurate transition models, calculating future state uncertainty, embedding high-dimensional observations into low-dimensional hidden states, etc. Hence, in what follows, we discuss the necessary theory and practical work being done in the field of model-based RL.

2.2.1 Preliminaries

We start defining the general notions of model-based RL with a POMDP. Although most of the literature assumes fully-observable MDPs, we will describe POMDP as a means to introduce a more general problem formulation, which will also be useful in Section 2.3 for the formulation of active inference.

POMDP can be formally defined as a tuple $(\mathcal{S}, \mathcal{A}, \Omega, \mathcal{P}, \mathcal{R}, O, b_0)$, where:

- \mathcal{S} is a set of hidden states;
- \mathcal{A} is a set of actions;
- Ω is a set of observations;
- \mathcal{P} is a set of transition probabilities;
- \mathcal{R} is a set of rewards, such that $\mathcal{R} : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$
- O is a set of observation probabilities
- b_0 is an initial belief state, defining a probability distribution over a hidden state.

More practically, an agent in an environment at time t receives an observation, $o_t \in \mathcal{O}$. The true state of the environment, $s_t \in \mathcal{S}$, is not observable and thus must be inferred by the agent. This uncertainty is represented as a probability distribution over states, more commonly referred to as belief state, $b(s_t)$. The agent then chooses to take an action, $a_t \in \mathcal{A}$, which transitions it to the next state, s_{t+1} , according to the transition probabilities, $p(s_{t+1}|s_t, a_t) \in \mathcal{P}$. The reward, $r_{t+1} \in \mathcal{R}$, is then retrieved from the state-action pair. After transitioning to the next state, the agent receives a new observation, o_{t+1} , and the cycle continues. Figure 2 shows a simple graphical model of a POMDP. The goal of RL is then to learn a policy, $\pi : \mathcal{S} \mapsto \mathcal{A}$, that maximises the expected discounted reward, $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t] | \pi$, where γ is the discounting factor which controls how myopic the agent is. Most commonly, the reward function is specified, although some works aim to learn it from samples [15].

Importantly, the distinguishing component of *model-based* RL is the transition dynamics model $p(s_{t+1}|s_t, a_t)$, which can be used, for instance, to augment a model-free Q-value

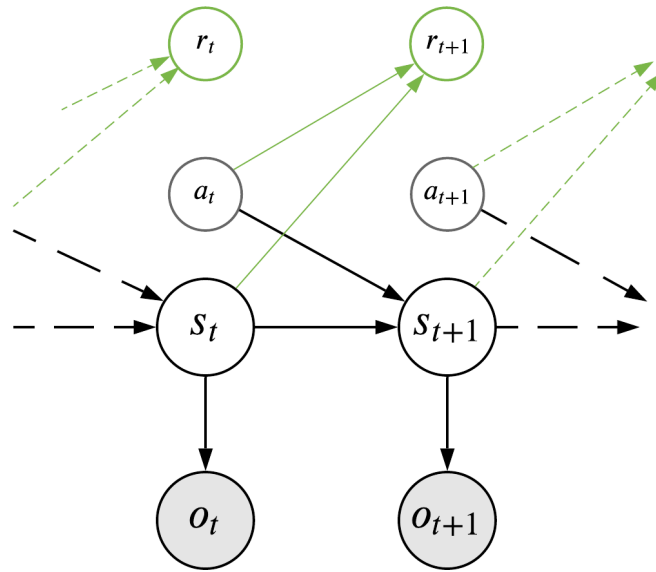


Figure 2: Simple schematic of a POMDP

estimator or perform planning into the future. This affords model-based RL several advantages over model-free approaches, which will be discussed next. Furthermore, such internal model of the world dynamics is central to the framework of active inference and is part of an agent’s generative model. Interestingly, just like the theory of evolution does not say anything about how the species must reproduce, the active inference framework does not specify how the generative model should be built. Hence, we must resort to clever engineering when dealing with modelling the transition dynamics, significant knowledge of which we can borrow from the fast-developing field of model-based RL.

2.2.2 Model-free vs. Model-based

We now briefly summarise the main performance differences between model-free and model-based RL. In the model-free approach, an agent is trying to directly map its observations of the environment to a policy without engaging in the task of learning a model of the world. These approaches have been compared and contrasted in a variety of studies, and the main distinctions include:

- Model-free RL is generally accepted as requiring a larger number of training samples, compared to model-based approaches [16];
- Model-free RL does not necessarily generalise well to novel tasks in the same environment [16];
- Model-free RL achieves better asymptotic performance in narrow tasks, given inaccuracies in the learned model for model-based methods (*model-bias*) [17];
- The learned transition dynamics can potentially be decoupled from any particular

task, meaning model-based methods afford their use on a number of tasks in the same environment [14].

Empirical evidence from neuroscience suggests that humans may well operate in both of these control modes, transitioning from one to another as the learning goes on [18]. This is also reflected in some studies from social sciences, such as Kahnemann’s famous formulation of humans’ fast (System 1) and slow (System 2) systems of decision-making [19]. A similar type of symbiosis of the two systems will be used in our baseline architecture in Section 3.

2.2.3 Review of Model-based RL

There are two most common directions, in which different model-based RL methods diverge from one another – in how the dynamics model is *learned* (representation and learning) and in how the transition dynamics is *used*. This section is structured by *what* methods are used to parametrise the dynamics model with important references to *how* the dynamics model is used in each respective work. Throughout, we aim to give an overview of the work that is being done in the field and, more importantly, the main problems facing it. By the end, we will draw important conclusions with respect to the identified problems and point out their relevance to this project.

Linear Models. Some works have explicitly modelled the transition dynamics using linear models and achieved quite good results, particularly in robotics [20, 21, 22, 23, 24, 25, 26]. In general, however, their application is limited to low-dimensional domains. Therefore, Watter et al. [21] propose to model the state dynamics with the use of latent state dynamics, which are encoded from the raw, high-dimensional images with the use of variational autoencoders (VAEs). Their system, E2C, exploits the local linearities of dynamical systems in the *feature space*, building up on the previously proposed frameworks like iLQG control [27] that are prevalent in robotics. However, it is stressed that these models do not perform well on more complicated non-linear model dynamics. Notably, encoding of high-dimensional observations into their respective low-dimensional latent representations has recently become a popular direction of research in model-based RL. It is also one of the most critical parts of the modern implementations of deep active inference agents, that similarly embed their observations into an internal latent space with VAEs (see Section 2.3.2 for more details).

Gaussian Processes. Similarly, Gaussian Processes (GPs) have been used to model transition models. In particular, their probabilistic nature can be used for state uncertainty estimation, which can be incorporated in the planning module to make more cautious predictions. Deisenroth and Rasmussen [28] introduced PICLO (Probabilistic Inference for Learning Control) that uses a GP dynamics model to train a policy only on the state predictions with low uncertainty. In the paper, the authors argue that a poor dynamics model (also referred to as *model-bias*) in conjunction with the absence of uncertainty quantification will ultimately yield wrong optimisation steps for the policy. Thus, PILCO used GPs to model the transition dynamics, which were able to capture prediction uncertainties generated by the model of the environment; however, PICLO did not incorporate temporal

correlations between subsequent states, as well as scaling issues due to GPs non-parametric nature. Similarly, Ko et al. [29] used GPs to model the transition dynamics with relative success in low-dimensional regimes.

An alternative and recently more prevalent way to parametrise transition models is with the use of neural networks. These are particularly attractive due to their recent proven success in a variety of domains, including deep reinforcement learning (DRL) [30], ability to deal with high-dimensional data, existence of methods for uncertainty quantification, and a wide range of architectures applicable to different types of data. Forward models implemented by neural networks generally fall into two camps – based on feed-forward multilayer perceptrons or on autoregressive models.

Feed-Forward Neural Networks. Up to this point, feed-forward networks are the most popular way to parametrise a transition dynamics model. Nagaband et al. [17] do exactly this to assist a model-free learner. Their transition model is used to sample K action sequences, for which the rewards are calculated, and actions with the highest reward are chosen. Using model-predictive control¹ (MPC), the actions are then continuously executed and re-evaluated in a closed-loop control loop. This style of planning is also possible in active inference, in which an agent would recalculate the value of the expected free energy after every new observation (refer to Section 2.3). However, this procedure suffers from high computational complexity and is generally wasteful for agents with a good generative model. Feinberg et al. [32] similarly use a neural network to parametrise the transition model. However, in their work, the authors attempt to take the best of both worlds by using a model-based approach to predict the rewards in the short-term, and model-free approach for long-term predictions. Their system was successful at reducing sample complexity in a number of OpenAI Gym [33] environment analogues with continuous state-spaces. However, these analogues were fully-observable, and the state-space of the tasks was limited to relatively low dimensions.

Alongside the works that simply use feed-forward networks for the parametrisation of transition models, significant amount of research has also been dedicated to how they can be used further to quantify state uncertainty. As mentioned, this can be beneficial for dealing with model-bias and sequential error accumulation. Crucially, in active inference, an agent’s generative model is intrinsically probabilistic, thus requiring methods that allow for uncertainty quantification. In model-based RL, one of the most widespread and straight-forward techniques for this is ensemble methods. For instance, Buckman et al. [34] introduced an extension to the aforementioned work by Feinberg et al. [32] by incorporating uncertainty of the roll-out predictions (imagination) using ensembles. This allowed the algorithm to adaptively sample roll-outs of different lengths, depending on prediction uncertainties. The algorithm showed improved sample efficiency on several low-dimensional tasks in the Ope-

¹This refers to the idea that an agent should i) plan its action according to the internal model of the world, ii) execute the action, iii) observe the effects of its actions, and finally iv) re-plan the next action again. This essentially means that an agent has a closed-loop feedback system that allows it to correct for the inaccuracies in its model, as well as environment’s and its actions’ intrinsic stochasticity. MPC has been used before to overcome model-bias [31], but it suffers from low credit assignment performance and high computational cost.

nAI Gym. Several other works [35, 36, 37] also use an ensemble of networks for modelling world dynamics to overcome overfitting and model-bias. Specifically, Kurutach et al. [35] use an ensemble of transition models to avoid reliance on any one of them, where as Kalweit et al. [36] augment a popular Deep Deterministic Policy Gradient (DDPG) algorithm [38] with artificially-generated roll-outs and adjust the proportion of imagined roll-outs for training depending on the degree of uncertainty. Further, Clavera et al. [37] use the transition model to parametrise a diagonal Gaussian, thus taking into account both epistemic (difference in predictions between ensemble networks) and aleatoric (entropy of the parametrised diagonal Gaussian) uncertainty. This technique of parametrisation of a *belief* state resembles a number of current deep active inference works, where the dynamics model is used to transition from one belief state to another given an action (see Section 2.3.2). Yet, we stress that all of the afore-mentioned works are still characterised by their relatively simple testing environments (OpenAI Gym), which puts into question the scope of their application in more complex testbeds such as the Animal-AI environment.

Bayesian Neural Networks. More sophisticated approaches include Bayesian neural networks (BNNs) and approximate inference techniques. One such example is Deep PILCO [39], in which the earlier insights about the connection of Monte Carlo (MC) dropout in neural networks – as being an approximation to a Gaussian Process – provided a way to quantify prediction uncertainty [40]. Nowadays, using MC dropout for uncertainty quantification in neural networks is one of the most commonly used implementations of BNNs due to its simplicity and effectiveness; however, some works show evidence of its unsatisfactory performance for uncertainty quantification [41]. Nevertheless, we employ this method in our work, see Section 3. More recently, Depeweg et al. [42] used an implementation of BNNs with the use of α -divergence minimisation, in order to construct an approximation to the transition dynamics. In their approach, however, the captured uncertainties are used to sample *weights* in the transition BNN for a given action (retrieved from a separate deterministic, policy network). Based on these sampled weights, their algorithm can then get an estimated expected cost of any particular transition trajectory (according to some specified cost function). Assuming that policy, model, and cost function are differentiable, the algorithm can use stochastic gradient descent to update the policy network parameters. The work showed improvement to the standard BNNs, enabling uncertainty quantification over more complex stochastic functions, and presents itself as a great future alternative to MC dropout for our system.

Segment-Based Approaches. Another potential way to deal with temporal error accumulation is to predict entire trajectories at once. For instance, Mishra et al. [14] employ a segment-based approach for predicting the environment dynamics – rather than create Markov-based predictions (one state at a time conditioned on a state in the previous time-step), the authors make predictions for the entire segments. This is done with the use of conditional variational autoencoders and autoregressive decoders. In similar fashion, the paper encodes policies – generating ‘latent action priors’ – which allow the algorithm to then sample future actions from the trained conditional decoder. This is aimed at tackling the issue of agent exploitation of the model-bias, which forces actions that achieve high reward

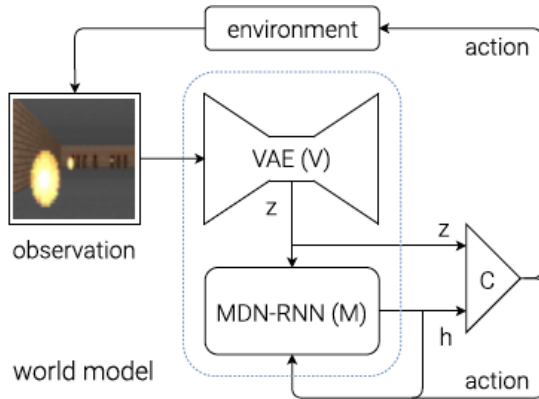
under an inaccurate learned transition dynamics.

Meta-Learning. Further approaches include meta-learning methods, which have recently been increasing in popularity. Authors of MB-MPO [43] propose a model-based approach based on meta-learning that does not strictly rely on the accuracy of agent’s internal model. Instead, they employ an ensemble of learned forward models and train a policy that can adapt to any of the models in one gradient step. The authors argue that although ensemble methods for model uncertainty are useful for the reduction of model-bias, it is not sufficient for more complicated domains, where, as they argue, meta-learning is necessary. Authors point to some advantages of this approach: the one-step constraint regularises the learned policy, encourages exploration by exploiting each of the ensemble model deficiencies, and leads to faster adaptation to different tasks. However, the proposed method suffers from a relatively high computational complexity, given that several transition models need to be trained at once. This significantly limits its use in sufficiently complex environments.

Autoregressive Models. Significant attention has also been directed to transition models that attempt to bypass Markovian assumptions with the use of autoregressive models, such as recurrent neural networks (RNNs). In a paper by Ha et al. [6], authors describe an approach to model the transition dynamics with a Long Short-Term Memory (LSTM) [44] networks over the latent states of the environment (encoded by a pre-trained VAE), see Figure 3. Here, however, the transition model is used *only* to encode temporal information into the hidden state of an LSTM, which is later used to train a policy network. Although the use of LSTMs is a step forward to learning temporal relations between events, they are known to be slow at training – especially in environments where the change in state given an action is small. Similarly, LSTMs are still limited in their ability to learn relations between temporally-distant events, which is similarly exacerbated in environments where little change occurs given an action. The authors also point out that using a VAE (to encode the observed states into a latent representation) independently off the policy and the transition model has a disadvantage, as the VAE would encode irrelevant information that would not be useful for solving a task. They also stress on the importance of incorporating curiosity-based exploration in more complex environments in the future work, so that the agent keeps exploring and refining its model. All of these points will be addressed in this project using the framework of active inference, a prioritised replay buffer (Section 4), and a temporal episodic memory module (Section 5). For instance, as will be discussed later on, active inference intrinsically includes an exploration component in its mathematical formulation, allowing the agent to keep exploring its environment, even at test time. Furthermore, Section 4 will introduce PRB that will be used to sample training data conducive to learning more *action-centric* representations. Lastly, Section 5 will describe a novel way to train an autoregressive transition model on sequences of selectively picked states, which would allow the model to bypass the problem of slow environment dynamics and more effectively learn dependencies between temporally-distant events.

Another recently-proposed system, I2A [16], used an autoregressive transition model to augment learning with imagination-based approach. Specifically, it used a hybrid approach with a model-based component complementing the learning of a model-free component.

Recent work by Ke et al. [45] similarly employs a latent dynamics model using an LSTM. Additionally, the authors implement a technique to drive the latent states to carry useful information for longer-term roll-out predictions. They achieve it by introducing a regularising auxiliary cost with respect to the encodings of future observations. The system showed improvement in sample complexity and total reward collected against a recurrent policy algorithm on a number of OpenAI Gym tasks.



```

def rollout(controller):
    ''' env, rnn, vae are '''
    ''' global variables '''
    obs = env.reset()
    h = rnn.initial_state()
    done = False
    cumulative_reward = 0
    while not done:
        z = vae.encode(obs)
        a = controller.action([z, h])
        obs, reward, done = env.step(a)
        cumulative_reward += reward
        h = rnn.forward([a, z, h])
    return cumulative_reward

```

Figure 3: Implementation from Ha et al. [6]: i) observation is encoded into a latent state, z , via a VAE (V); ii) autoregressive transition (M) model encodes a sequence of observed states until the present time; iii) action-selector (C) produces an action.

Some notable mentions also include hierarchical RL [46], Dyna algorithm SLBO [47], Guided Policy Search [48], Stochastic Value Gradients [49], and the so-called shooting algorithms RS [50], PETS-RS and PETS-CEM [51]. Several authors have also proposed adaptive models [52, 53, 54].

2.2.4 Summary and Problem Identification

To summarise, we reviewed a range of methods used to parametrise transition models and quantify state uncertainty. Specifically, we saw that GPs and linear time-varying models show excellent performance in low-dimensions but are unsuccessful when it comes to more complex or high-dimensional environments. Therefore, our review largely focused on deep learning methods that offer a suite of methods allowing for usage in high dimensions. These methods have demonstrated their effectiveness in a range of environments with relatively high complexity. Furthermore, several techniques have been used to quantify state uncertainties in neural networks, such as ensemble methods and BNNs. However, we noted that the former is computationally expensive, while the latter is still an active area of research. Within the deep learning framework, we highlighted autoregressive methods, which have been rising in popularity due their ability to work in non-Markovian environments and learn temporal dependencies between states.

As discussed, another distinguishing component of the reviewed papers was how the dynamics model was used. This can generally be categorised into four main groups: i) model-based planner with a model-free action proposer; ii) data augmentation for a Q-function learner

using imagination roll-outs; iii) information gained from roll-outs used as input to a policy; iv) purely model-based approach (e.g. model-predictive control).

The important conclusion that becomes evident after the review is that model-bias and error accumulation are still a major challenge and focus of the research community in model-based RL. In particular, it is clear that few of the current models attempt to learn long-term temporal relations between events, which is crucial in environments characterised by the presence of important, temporally-distant events and sparse rewards. Interestingly, the afore-discussed models all use the objective (ground-truth) timescale of their environments to construct an agent’s internal dynamics model. As part of this project, we will try to answer whether an agent can *learn* a more useful and **subjective** timescale of the world, and thus be able to predict further into the future and with less computational resources.

2.3 Active Inference

2.3.1 Preliminaries

Active inference is a corollary of the free-energy principle proposed by Karl Friston [1]. In this framework, an agent embedded in an environment aims to do two things: i) minimise surprisal from the observations of the environment under the agent's internal model of this environment, and ii) perform actions so as to minimise the expected surprisal in the future.

To begin a more formal definition, it is relevant to draw a distinction between the concepts of a *generative process* and a *generative model*. While the former describes the true generative dynamics by which the observations are generated in an environment, the latter refers solely to the internal model that an agent possesses of that environment. In the context of neurobiology, the distinction implies that given that the true state of the environment, \bar{s} , is not attainable, the brain constructs an internal generative model to explain its sensory inputs, o_t . Figure 4 shows a simplified graphical model of this distinction.

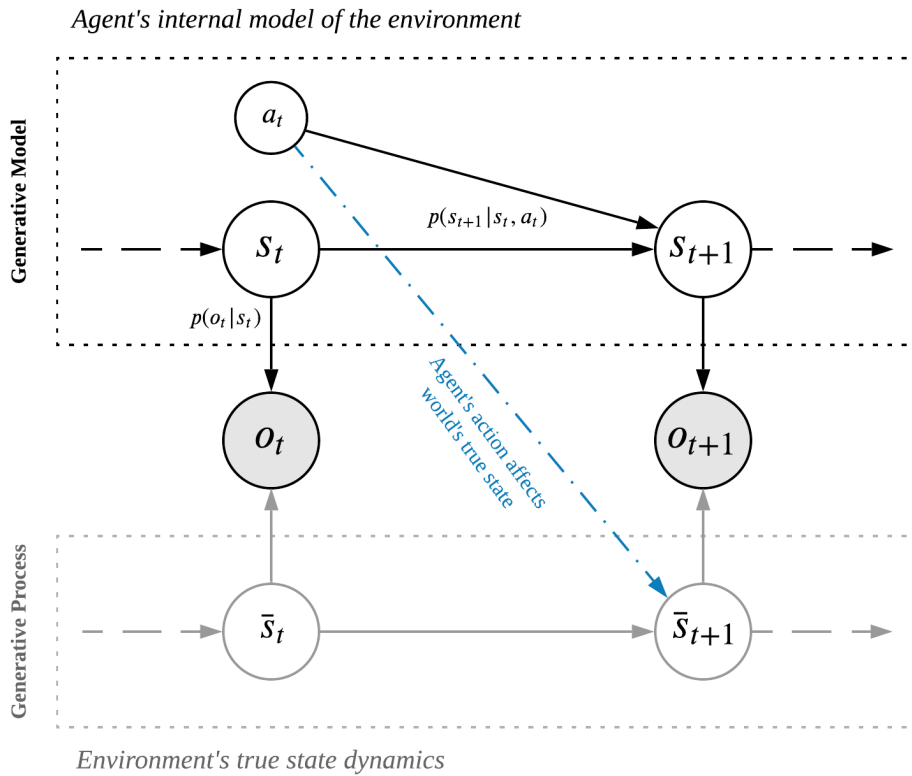


Figure 4: **Distinction between a *generative process* and a *generative model*.** Here, \bar{s}_t is the true hidden state of the environment, s_t is the state of the environment inferred by the agent, o_t are observable sensory inputs, and a_t is an action taken by the agent. In this case, we also note that the generative process is defined as a partially-observable Markov decision process.

We now proceed to the formal definition. First, we will cover objective (i) – the minimisation of surprise under the agent’s *generative model* of the environment. We start by defining the surprise as

$$-\log P(o) \tag{1}$$

where $P(o)$ is the probability of observing observation o . This quantity has various names in different fields of science, such as negative *model evidence* or *marginal likelihood*. We can immediately observe that the minimisation of this quantity leads to the maximisation of model evidence, which intuitively implies that the agent aims to perfect its generative model at explaining the sensory observations from the environment. From the definition of the generative model in Figure 4, an agent holds that the observations are produced by some hidden state s , referring to the hidden state that is part of the agent’s generative model. We can thus write the marginalised joint distribution as:

$$-\log P(o) = -\log \int_{s \in S} P(o, s) ds \tag{2}$$

This integral is intractable (in large state spaces), so we have to resort to variational inference by introducing an approximate posterior distribution $Q(s; \theta)$ parametrised by θ , such that:

$$-\log P(o) = -\log \int_{s \in S} \frac{P(o, s)}{Q(s)} Q(s) ds \tag{3}$$

Using Jensen’s inequality, we can bring the log inside the integral:

$$-\log P(o) \leq -\int_{s \in S} Q(s) \log \frac{P(o, s)}{Q(s)} ds \tag{4}$$

Now, we can flip the sign and write the integral as an expectation, which is the variational free energy (\mathcal{F}):

$$-\log P(o) \leq \mathcal{F} = \mathbb{E}_{Q(s)} \left[\log \frac{Q(s)}{P(o, s)} \right] \tag{5}$$

We can see that the expectation upper-bounds the surprise by re-arranging the terms:

$$-\log P(o) \leq KL [Q(s) || P(s|o)] - \log P(o) \tag{6}$$

since KL-divergence is a non-negative quantity. Similarly, this expression clearly shows that the free energy is minimised when the approximate posterior, $Q(s)$, becomes the true posterior, $P(s|o)$. Furthermore, we can decompose the expression alternatively as,

$$-\log P(o) \leq \underbrace{KL [Q(s) || P(s)]}_{complexity} - \underbrace{\mathbb{E}_{Q(s)} [\log P(o|s)]}_{accuracy} \tag{7}$$

This shows that the minimisation of the free energy also means minimising the *complexity*² of *accurate* explanations for the observations, thus realising a form of Occam’s razor.

²Bits of information encoded in the approximate posterior that are not in the prior

As mentioned before, the objective is to minimise the variational free energy \mathcal{F} – which is equivalent to maximising model evidence and minimising surprise of an observation. How can an agent achieve this? There are only *two* ways, in which an agent has control over the minimisation of this quantity – to *adjust the parameters* of its generative model or to *actively sample* observations that minimise surprise under its generative model by taking actions in the environment. So far, however, we have not introduced action and temporal dynamics into the objective. Thus, we come to objective (ii), which states that an agent also tries to minimise surprise (\mathcal{F}) from its observations in the future. This, in turn, implies that an agent must possess prior beliefs about its actions. The formulation of this prior belief is often explained using *reductio ad absurdum* argument [3], which leads to the conclusion that if this prior belief realises the objective of minimising the free energy, then the prior belief itself must be that the action will minimise the free energy. Otherwise, a free-energy minimising agent would take actions that do not minimise the free energy, which is a contradiction.

Thus, we introduce the temporal component and extend the minimisation of surprise into the future by considering the *expected* variational free energy (EFE) [2],

$$G(\pi, \tau) = \mathbb{E}_{P(o_\tau | s_\tau)} \left[\underbrace{\mathbb{E}_{Q(s_\tau | \pi)} [\log Q(s_\tau | \pi) - \log P(o_\tau, s_\tau | \pi)]}_{\text{variational free energy, } \mathcal{F}} \right] \quad (8)$$

where we additionally take an expectation over the free energy with respect to future observations at time τ and condition our distributions on policy π . The free-energy minimising system must, therefore, *imagine* the future observations given the policy and calculate the expected free energy conditioned on taking this policy. Then, actions that led to lower values of the EFE are chosen with higher probability, as opposed to actions that led to higher values of EFE. Hence, we employ a softmax function, $\sigma()$, to define the distribution, from which an action can be sampled:

$$P(\pi) = \sigma(-\gamma G(\pi)) \quad (9)$$

where $G(\pi) = \sum_{\tau > t} G(\pi, \tau)$, γ is the temperature parameter, and t is the present time-step. Here, we again see that sampling actions from this distribution would imply that an agent would act so as to minimise the expected free energy across time.

At this point, a natural question arises – how does one incorporate agent goals in this framework? By inspecting the described formulations closely, we see that the agent will aim to act such that it observes what it's *expecting to see*, i.e. the prior over the observations, $P(o)$. Put simply, these observations would lead to the lowest values of surprise, the value of which the agent is trying to minimise. Hence, the goals are encoded as *preferences* over observations. An example of such a prior for the Animal-AI environment can be seen in Figure 5.

The formulation of active inference provides an elegant solution for the infamous exploration/exploitation trade-off – an agent is to indulge in exploitative behaviour under its model to minimise surprise, while also exploring the environment to avoid being surprised

in the future [55]. More formally, Eq. 8 can be decomposed into,

$$G(\pi, \tau) = \mathbb{E}_{P(o_\tau|s_\tau)} [\mathbb{E}_{Q(s_\tau|\pi)} [\log Q(s_\tau|\pi) - \log P(o_\tau, s_\tau|\pi)]] \quad (10)$$

$$= \mathbb{E}_{P(o_\tau|s_\tau)} [\mathbb{E}_{Q(s_\tau|\pi)} [\log Q(s_\tau|\pi) - \log P(o_\tau) - \log P(s_\tau|o_\tau, \pi)]] \quad (11)$$

$$= - \underbrace{\mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)} [\log P(o_\tau)]}_{\text{extrinsic value}} - \underbrace{\mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)} [\log P(s_\tau|o_\tau, \pi) - \log Q(s_\tau|\pi)]}_{\text{epistemic value}} \quad (12)$$

Expression 12 shows that low expected free energies encourage the agent to resolve uncertainty about states (referring to *epistemic value*) and take actions to increase the likelihood of the observational prior (referring to *extrinsic value*) under its generative model. We can further re-arrange the EFE by considering the Bayes rule and treating $Q(s_\tau|\pi)$ as a prior and $P(s_\tau|o_\tau, \pi)$ as a posterior, such that:

$$P(s_\tau|o_\tau, \pi) = \frac{P(o_\tau|s_\tau, \pi)Q(s_\tau|\pi)}{Q(o_\tau|\pi)}. \quad (13)$$

This can now be substituted into expression 12,

$$\begin{aligned} G(\pi, \tau) &= - \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)} [\log P(o_\tau)] \\ &\quad - \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)} [\log P(o_\tau|s_\tau, \pi) - \log Q(o_\tau|\pi)] \end{aligned} \quad (14)$$

Now, we merge the expectations and split them again in a different way,

$$G(\pi, \tau) = \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)} [\log Q(o_\tau|\pi) - \log P(o_\tau) - \log P(o_\tau|s_\tau, \pi)] \quad (15)$$

$$= D_{\text{KL}} [Q(o_\tau|\pi) || P(o_\tau)] - \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)} [\log P(o_\tau|s_\tau, \pi)] \quad (16)$$

$$= \underbrace{D_{\text{KL}} [Q(o_\tau|\pi) || P(o_\tau)]}_{\text{expected cost}} + \underbrace{\mathbb{E}_{Q(s_\tau|\pi)} [H [P(o_\tau|s_\tau, \pi)]]}_{\text{expected ambiguity}}, \quad (17)$$

where H denotes Shannon entropy.

Eq. 17 splits into two meaningful terms. The *expected cost* measures the divergence between an agent's preferred prior observation, $P(o_\tau)$, and the expected policy-conditioned observations, $Q(o_\tau|\pi)$. Thus, the agent would prefer policies that produce observations more similar to the prior. The *expected ambiguity* quantifies the predicted uncertainty about the outcomes given a policy. This means that an agent would prefer policies that lead to more informative observations [56].

In this project, we will be creating an active inference agent that parametrises its generative model using deep learning methods. The following literature review is, thus, largely focused on *deep* active inference literature.

2.3.2 Review of Active Inference Literature

Up to this point, most of the work on Active Inference has been done in low-dimensional and discrete state spaces [57, 56, 58, 59]. Recently, however, there has been a rising interest on applying active inference to environments with continuous and/or large state spaces.

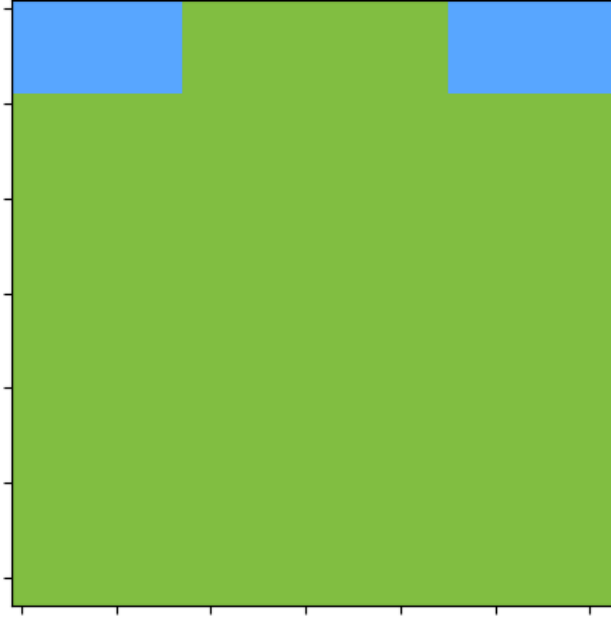


Figure 5: Example of an observational prior, $P(o)$. An agent in the Animal-AI environment can be made to *expect* to see this image. Thus, in the process of surprise minimisation, it will tend to go towards a green reward.

A work by Ueltzhöffer [60] was one of the earliest papers to implement active inference with the use of deep learning techniques. There are a few key components of this implementation. First, the distribution of the next state s_{t+1} is modelled as a factorised Gaussian, whose sufficient statistics is calculated via a neural network with an input of a preceding state s_t . Similarly, the distribution of observations, o_t , conditioned on the latent state, s_t , take the form of a mean-field Gaussian. Second, the variational posterior density of state s_t conditioned on current observations and the previous state s_{t-1} is approximated using amortized inference with a diagonal Gaussian factorisation. The policy is retrieved by training a separate neural network $p_\theta(a_t|s_t)$. This implementation was applied to the Mountain Car problem [61] by instilling the observational prior of being in the correct state of the environment – similarly to the prior explained in Figure 5. Although this work was first at merging deep learning and active inference, it has significant limitations such as the agent having access to the underlying physics of the environment and the apparent simplicity of the testbed.

Catal et al. [62] has a similar implementation approach to Ueltzhöffer [60] with a few modifications. One is the use of a habit network, which calculates an agent’s policy and which is minimised with respect to the expected free energy. This habit network is also used to *sample* actions for the evaluation of the expected free energy. As a result, this paper provided first inklings for the integration of model-free components into a purely model-based active inference agent. This symbiosis will similarly be used in our system (Section 3).

Millidge [63], too, has very similar basic components in his implementation. One novelty is the calculation of the expected free energy by bootstrapping with the use of a separate EFE network (which maps a state-action pair to EFE). The author points to the limitation

of this approach, as the approximated values of EFE can be inaccurate, thus reducing the effectiveness of bootstrapping. Further, they mention optimisation for the precision parameter (used for selection of a policy) as a potential avenue for future improvements. Having used OpenAI Gym environments, Millidge emphasises that the underlying dynamics is still relatively simple. Because of this, he attributed a limited effect of the epistemic uncertainty term on the performance of the model, and suggests experiments in more complex environments.

The research community is slowly advancing their deep active inference agents. For instance, a recent work by Tschantz and colleagues [55] tested their system in continuous and relatively high-dimensional state spaces, and implemented Bayesian neural networks to quantify model uncertainty. Specifically, BNNs were implemented using Bayes by Backprop technique [64]. The authors also used amortized inference for learning latent state representations. BNNs and amortized inference proved suitable for the calculation of EFE, and our work similarly employs these two techniques.

Many works specifically refer to the distinct differences between the frameworks of RL and active inference. A comprehensive review of this was recently published by Sajid et al. [4]. In this paper, the authors use a very simple OpenAI Gym environment to compare Q-learning with active inference and, importantly, contrast their approaches, mentioning:

- *Reward engineering.* As mentioned in Section 2.2, the goal of an RL agent is to maximise the expected discounted rewards. These rewards, however, need to be formulated explicitly. Moreover, in the absence of any reward signal, a reinforcement learning agent would have no means of training. Reward engineering can be difficult and may lead to an agent learning a sub-optimal policy, if formulated erroneously. On the other hand, in active inference, rewards are encoded as preferred outcomes under an agent’s prior probability distribution over its observations. Thus, the agent acts so as to observe what was encoded as a preference in its prior. Moreover, because of this representation, it is possible to learn these prior preferences by placing a distribution over them and letting the agent interact with the environment, as demonstrated in Sajid et al. [4].
- *Exploration vs. Exploitation.* How does one deal with exploration-exploitation trade-off in reinforcement learning? Currently, many methods have been proposed based on some form of weighted intrinsic reward, whose weight must be tuned. In contrast to the ad-hoc approaches to curiosity in RL [65, 66] – such as policy entropy [67], state entropy [68], information gain [69], prediction error [70], ensemble divergence [71], empowerment [72] – it arises naturally from the mathematical formulation of the free energy.
- *Belief over vs. Value of a state.* This is commonly emphasised as one of the key distinctions between the RL and active inference frameworks. In RL, we are primarily interested in optimising some *value* function for a given state, where as in active inference it is the free energy functional over an agent’s *belief* about the state.

In Tschantz et al. [73], the authors explored the use of active inference for the creation of action-oriented models of the world. The basic idea is that an agent’s internal model must be intrinsically connected to the actions that the agent can take. Similarly to what was suggested in Ha et al. [6] (learning action-centric latent encodings), the authors suggest using the expected free energy to induce representations that do not necessarily encode a more accurate representation of the reality, but rather ones that encode more useful features related to goal-directed behaviour. As a result, they found that EFE approach learns less accurate models (which are *action-conditioned*) than epistemic/random exploration approaches, yet performs better in a variety of tasks. This paper discusses a rather important problem, which was also recently questioned by Freeman et al. [74] in a model-based RL setting. Specifically, does an agent require a perfect model of the environment to make sensible decisions? In this project, we partly address this problem by training our generative model with samples that are intrinsically linked to action and the highest number of useful affordances for our agent (Section 4).

Some works have also attempted to investigate the original formulation of the *expected* free energy and instead propose alternative objective functions. Tschantz et al. [75] implements an active inference agent with the use of the free energy of the expected future. This new objective function was derived as a more representative definition of the generalisation of the free energy functional when projected to future states [76]. Alternative objective functions (specifically, the intrinsic motivation component) for active inference were also studied by Depeweg et al. [42].

Very recently, Fountas et al. [10] introduced a novel deep active inference system with the use of Monte Carlo tree search (MCTS). Specifically, the agent is equipped with a POMDP generative model, which consists of the following components: i) transition model $P(s_{t+1}|s_t, a_t)$, ii) state encoder/decoder $Q(s_t|o_t)$ and $P(o_t|s_t)$, and iii) action predictor $Q(a_t|s_t)$. All of the mentioned conditional probability distributions have closed-form formulations (Gaussian- and Bernoulli-distributed) and are parametrised using Bayesian neural networks with the use of MC dropout [40] and amortized inference [77].

Importantly, the paper tackles the question of efficient EFE calculation using MCTS. As explained in Section 2.3, action-selection is achieved by calculating the EFE values for different policies and later sampling actions proportional to the negative EFEs. However, this process is computationally expensive, given that the number of possible paths increase exponentially with every step an agent takes. Fountas et al. [10] address this issue by combining MCTS with the action predictor $P(a_t|s_t)$ (which is approximated by the variational posterior, $Q(a_t)$). Namely, MCTS generates a weighted tree that is used to sample policies from the current time-step, where the weights refer to the agent’s estimation of the EFE given a state-action pair. A more detailed explanation of the architectural details and the modified version of this model can be found in Section 3.

The proposed architecture showed promising results in the Animal-AI environment by learning a satisfactory generative model and learning to retrieve the reward on a ‘variation of preferences’ task. Yet, we must point to two main issues facing this architecture:

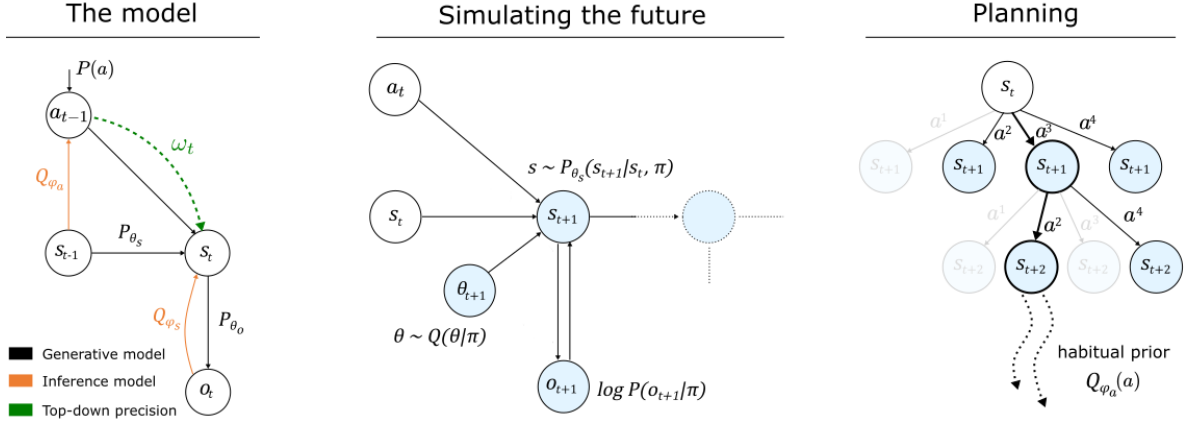


Figure 6: **Architecture of the deep active inference agent with MCTS by Fountas et al. [10].** *The model:* shows the graphical model of random variables for how observations are produced. *Simulating the future:* shows how the future observations are generated. *Planning:* shows the MC tree search with the use of the habitual prior.

1. The agent’s transition model was trained on every third step (i.e. given a single chosen action, an agent would take three such actions in a row – and only then retrieve an observation). This was done to improve the agent’s transition model at predicting states farther into the future. As discussed, this is crucial for calculating EFE values that are representative of the long-term future and, as a consequence, for taking more sensible actions. Predicting further into the future was also extensively discussed in Section 2.3.2, where the problems pertaining to learning a good transition model for temporally-distant events is still an active area of research in model-based RL. In this project, we will address this issue by asking the following question: can we come up with a principled way to perform temporal jumps (of arbitrary lengths) by equipping an active inference agent with means of defining a *subjective* timescale, over which the transition dynamics model would be trained? In Section 5, we will attempt to answer this question with the use of an external episodic memory module and a separate autoregressive transition model trained over the memories.
2. The current implementation included an on-line learning approach. Specifically, an agent would be spawned in $N \approx 50$ environments, sending back 50 observations used as a batch to train the generative model. Although this approach produces satisfactory results, there are three issues associated with it: i) subsequent observations are highly correlated data points, ii) observations are discarded after one training iteration, and iii) the samples contained in each batch may not be the best samples to train the neural networks on. With respect to (i), we stress that temporally-correlated data is generally considered to be detrimental for training neural networks [78]. With respect to (ii), observations that are *rarely encountered* in an environment are discarded in on-line learning and are used for training *only* when visited again. Moreover, the generative model may learn some states better than others, yet the frequency of their appearance in a training batch is uncontrolled, meaning that the network can be training on transitions that have been learned well yet rarely encountering poorly-predicted ones.

Finally, issue (iii) emphasises that an on-line agent is largely constrained by its current position in an environment to sample data and, thus, has very limited control over its training batches. These issues are addressed in this project with the use of a Prioritised Replay Buffer (PRB) [78], see Section 4.

Nevertheless, this paper represents the state-of-the-art system in the framework of deep active inference and serves as the foundation for this project. The use of MCTS presents itself as biologically plausible and efficient way for calculating EFE along distant trajectories [10]. As mentioned, the issues outlined above will be tackled with the use of an external episodic memory module and a prioritised replay buffer, which are both systems based on *episodic memory*. Because of this, the following section will provide a review on episodic memory in reinforcement learning and neuroscience.

2.4 Episodic Memory

2.4.1 Background and Motivation

Memory has been largely discussed in the neuroscience, cognitive science and artificial intelligence communities as one of the necessary components of an intelligent system. In neuroscience, human memory is usually partitioned into declarative/procedural and short-/long-term [7]. Declarative memory refers to information relating to facts or events that can be consciously recalled (declared) and is further categorised into semantic (factual information) and episodic (personal experiences). Procedural memory is predominantly used to describe unconscious memory, such as knowing how to ride a bike or type on a keyboard.

In the context of RL, procedural memory has been encapsulated by learning the value functions in model-free RL and semantic declarative memory by learning the world (forward) models [79]. The presence of these two types of controllers in a human brain has been studied in neuroscience and psychology [80], with strong empirical evidence that humans transition from model-based to model-free control in the process of learning and accumulation of experience. For instance, learning how to play guitar starts with a strong directed control (model-based) and, with experience, becomes intuitive and effortless (model-free).

Using forward models when dealing with sequential decision-making imposes a high computational burden on the semantic memory [18], while also being prone to large errors as a result of noise or prediction uncertainty. This problem was largely elucidated in Section 2.2, where a number of proposed model-based algorithms attempt to deal with problems related to *model-bias*, sequential accumulation of prediction error, and long-term predictions. Accordingly, there results a need for a different, complementary approach to deal with this issue.

Here, *episodic memory* is of particular interest, because of its deep temporal nature that may allow agents to learn long-term dependencies [79]. Episodic memory is used to describe autobiographical memories that link a collection of first-person sensory experiences at a specific time and place [81], such as remembering your last birthday party or your first day at university. Past works from neuroscience provide evidence that episodic memory plays an important role in human learning [7], capturing a wide range of potential functional purposes, including assisting in the formation of semantic declarative memories [9] or in the construction of novel future scenarios [82, 83]. In what follows, we review the applications of episodic memory in the context of model-based RL, as well as some of the recent works from neuroscience that informed the design of our system in Section 5.

2.4.2 Review of Episodic Memory in RL Literature

In 2008, a paper published by Lengyel et al. [18] has emphasised the issues with forward models outlined above and proposed a simple solution based on the use of episodic memory – *instance-based control*. In that work, the authors compared model-based approach to the proposed episodic control in scenarios where very little experience of the environment is available. Essentially, the model would record states and actions that resulted in a sufficiently large reward, and would later engage in an exploitative behaviour if the state is

encountered again. The paper showed the importance of incorporating episodic control as a low-data regime alternative to model-free and model-based approaches.

More recently, a primitive sort of memory has also been incorporated into model-free deep reinforcement learning systems [84, 85] in a form of a buffer that stores tuples of experience. These experience tuples are then continuously used in a mini-batch learning (also termed *replay*) to train deep neural networks parametrising Q-value functions. These buffers, however, do not incorporate any associative or temporal dependencies [86] between the memories, and are used primarily to stabilise on-line learning (which otherwise uses highly-correlated data for training and can therefore lead to *catastrophic forgetting*). Furthermore, Hansen et al. [86] argue that such use of episodic memory is slow and that important memories may be removed due to the limited size of the buffer. A work by Schaul et al. [87] develops on these ideas and uses a *prioritised* experience replay to sample experiences that produced larger prediction errors more frequently. Similarly, in this project we reconcile the technique of data prioritisation with the framework of active inference and further advance it by analysing the different prioritisation strategies that could be used for an agent embedded in a complex testing environment (Section 4).

Another idea, coined as Episodic RL, is to simply store the previous visited states in the memory bank along with an associated sum of discounted rewards after that state (or trajectories). The agent will then approximate state value as a weighted average of value functions of all the similar states³ [88, 79]. In the case where the states are similar, such value approximation can be made rapidly, even with just one-shot learning. This type of learning is empirically supported from the analysis of hippocampus [89, 90]. Evidence of instance-based learning in hippocampus has been theorised for some time [91, 92] and contrasts with the cortical learning that involves statistical summaries of the input [93, 94]. Such duality provides room for the integration of a fast-learning system that would complement the slower gradient-based learner. Gerschman et al. [79] points to several problems with this approach, however. First, the stored trajectories will tend to be short because of the discounted nature of the reward (myopic agents). Secondly, in continuous environments, the states are unlikely to be revisited; hence, an agent needs to be able to generalise and have some metric of state similarity.

Notable recent work by Blundell et al. [95] on Model-free Episodic Control (MFEC) tackles the problem of sample-inefficiency with the use of episodic memory. The intrinsic gradient-based updates that characterise neural network learning make changing the Q-value approximation slow, even for events that carry rich information. Therefore, MFEC attempts to solve this issue by having a separate fast learner that is based on the replay of action sequences from the current state. However, tabular records suffer from two main problems – memory constraints and state generalisation (states are almost never revisited in continuous environments). The former is tackled by removing memories that were least recently-updated. The latter is tackled with the use of k -nearest neighbour averaging, i.e.

³Here, similarity can be measured in a variety of ways and is usually defined as a distance metric in the kernel space [79]

taking k nearest neighbours of the state at hand and performing linear averaging. With these tools, Q-value function can be approximated for all states and be iteratively improved through time and experience. MFEC was further developed to a differentiable version, creating Neural Episodic Control (NEC) [96], which performed significantly better on a number of Atari games.

With regards to neural networks, Differential Turing Machines were one of the early incorporators of memory into a network-based learning system. They used a memory bank alongside an LSTM to augment the learning process, however struggled to scale to high-dimensional tasks [97]. Feedback Recurrent Memory Q-Network (FRMQN), along with two simpler architectures, were introduced by Junhyuk et al. [98]. This architecture employs external memory, to which it can write high-level features extracted from a neural network (i.e. encoded observation). It also employs a context vector for memory retrieval using soft attention mechanisms and action-value estimation.

Moving on, EVA [86] attempts to incorporate the trace information in the replay buffer, in order to construct an agent that can learn both slowly (using gradient-based updates of a DQN – parametric) and quickly (using value iteration within episodes by employing remembered traces – non-parametric). These two modes of Q-value calculation are then linearly combined to get one estimate. The non-parametric Q-value estimate is calculated with the use of trajectory-centric planning or kernel-based RL. The former makes use of the parametric value approximator for counter-factual actions (not encountered previously) when calculating the Q-value, while the latter is a generalisation of the former.

Zhu et al. [99] record episodic memory as the highest state-action pair for a given state. The states are transformed into a latent representation with the use of VAEs. Importantly, the novelty of their work comprises of the use of graphs to record memory contingencies and, thus, incorporating a form of associative memory. As part of the paper, they also introduced a value propagation algorithm, which is contrasted with trajectory-centric planning algorithm from EVA [86]. The main difference is that EVA can only propagate value through each particular episode, whereas the algorithm by Zhu et al. can propagate value between different episodes. However, like most episodic memory implementations every state-action pair is added to the buffer, which may be considered a waste of memory without sufficient justification.

Therefore, Jung et al. [100] more seriously consider the problem of limited memory and tackle the question of which memories to keep. Interestingly, they approach this problem by introducing a reinforcement learning system that optimises for a pre-defined reward and chooses which memories to keep in the buffer. Moreover, their memory system takes into account spatial and temporal dependencies, where the former is characterised by calculating the relative importance of a memory to the others and the latter takes into account the change of memory importance over time.

The use of (episodic) memory also closely touches on one-shot learning. In [101], authors motivate their work by pointing out the commonalities of rarely-occurring data points in

the training datasets. Although this work is largely focused on classifications tasks, their ad-hoc memory module could be incorporated into recurrent neural networks, making it directly suitable for modelling the world dynamics.

As can be seen, episodic memory can be used in a variety of ways to i) augment the performance of a gradient-based system, ii) perform instance-based control, and iii) one-shot learning. In this project, however, we will consider its alternative potential function. Namely, we design our system by considering episodic memory as events *worthy of remembering* and whose purpose is to assist agent’s ability to *imagine possible futures*. In neuroscience, the measure of *worthiness* of a memory is often studied in the context of the future, i.e. memories of the past are important because they assist us in making decisions in the present and ensuring future benefits [102]. In particular, some works have shown that patients with hippocampal lesions lose their ability to imagine the future [103] and some explicitly frame its function as to assist mental time-travel [104] (though precise functions of episodic memory are still subject to heavy debate [7]).

At the same time, recently, Fountas et al. [105] proposed a predictive processing model (equipped with a generative model) of *human time perception* with the use of episodic memories. In particular, it is known that humans do not perceive time evenly and do not possess an explicit clock mechanism responsible for keeping track of time [106]. As such, Fountas et al. devised a memory-based computational model that accurately accounted for the biases in human time perception, marking a potential direction for unifying time perception and episodic memory. Importantly, the criterion that the authors relied on to decide on the formation of an episodic memory was the *prediction error* produced by its generative model. The selection of this criterion is informed by the experimental evidence from neuroscience [107, 108, 109]. Similarly, in computer science, a work by Pertsch et al. [110] on discovering most useful (‘key’) frames in a video sequence for reconstruction and planning indicates that prediction error was one of the best criteria for deciding on such frames. By recording episodic memories based on this criterion, Fountas et al. argue that they were able to capture the *subjective timescale*, by which humans perceive time. The success of this episodic memory model has two implications for our project: i) it provides the basis for defining the active inference agent’s *subjective and more useful* timescale over which the transition dynamics can be learned, and ii) it informs the definition of the criterion by which episodic memories are recorded. More concretely, the problem of learning the transition model over longer time-steps (discussed in Sections 2.2.3 and 2.3.2) can effectively be addressed by ignoring the objective timescale of the environment and instead using the subjective timescale defined over recorded episodic memories. As we will see, the new subjective timescale allows the agent to vary how far into the future it predicts depending on the context it finds itself in.

2.5 Summarising Remarks

In conclusion, we reconcile the different parts of this literature review – model-based reinforcement learning, active inference, and episodic memory – and unify them in a single problem-solution formulation.

In the model-based RL section, we saw some of the most common problems pertaining to the field, namely model-bias and temporal prediction error accumulation, and discussed the state-of-the-art algorithms. Similarly, we saw that *all* of the proposed systems used the objective timescale of their environments to learn the transition dynamics, further exacerbating the algorithms’ sequential error accumulation and hindering their ability to predict further into the future. With respect to autoregressive models, this was also detrimental to their ability to learn long-term event dependencies, which is crucial in environments with sparse rewards. We stressed that these problems are still largely unaddressed.

In the active inference section, we introduced the basic formulations of the framework and similarly discussed the current state of the art in *deep* active inference. We noted that active inference almost entirely relies on the quality of the generative model, whose transition model is naturally susceptible to the same problems of model-bias and error accumulation. Therefore, we emphasised the need for a system that could learn long-term temporal dependencies between states and perform temporal jumps over unimportant intermediary states, whilst preserving the same qualities of a normal transition model (i.e. being able to learn the underlying physics of the environment).

The last section on episodic memory primarily described the current methods of incorporating episodic memory into model-based RL agents. It was found that most of these techniques either complement gradient-based systems or perform instance-based learning. At the same time, the neuroscience literature provided us with an alternative interpretation for the functional purposes of episodic memory; particularly, its use to assist a human in the imagination of novel future events and mental time-travel. Furthermore, we introduced a recently-proposed predictive processing system that successfully modelled human time perception with the use of episodic memories. Specifically, the work by Fountas et al. [105] used episodic memories to define a subjective timescale, which accounted for the underlying properties of human time perception. Inspired by this account, we will attempt to answer whether we can learn a more useful transition dynamics model trained over sequences of episodic memories.

In what follows, we will use these insights and implement two complementary memory modules on top of the baseline active inference agent (Section 3). First, in Section 4 we will reconcile the prioritised experience replay [78] with the framework of active inference in an attempt to improve the quality of the agent’s generative model. Second, in Section 5 we devise a novel way to learn an autoregressive transition dynamics model over the subjective timescale defined by episodic memories.

2.6 Note on Legal and Professional Considerations

This project deals with a range of theoretical concepts with no direct practical application or potential for misuse. We stress that this project does not make use of any sensitive biological data, nor does it involve human or animal participants. All of the training data has been synthetically generated using the Animal-AI environment. The systems devised over the course of this project have no connection to any human activities in the physical reality, do not relate to any military technology, and have no relation to environmentally harmful activities. All of the used and produced code is open-software and does not fall under any data protection laws.

3 Baseline Architecture

We take the deep active inference system devised by Fountas et al. [10] as the starting point with a few architectural and operational modifications. Following our discussion on active inference preliminaries in Section 2.3, we first need to define a POMDP generative model – Figure 7.

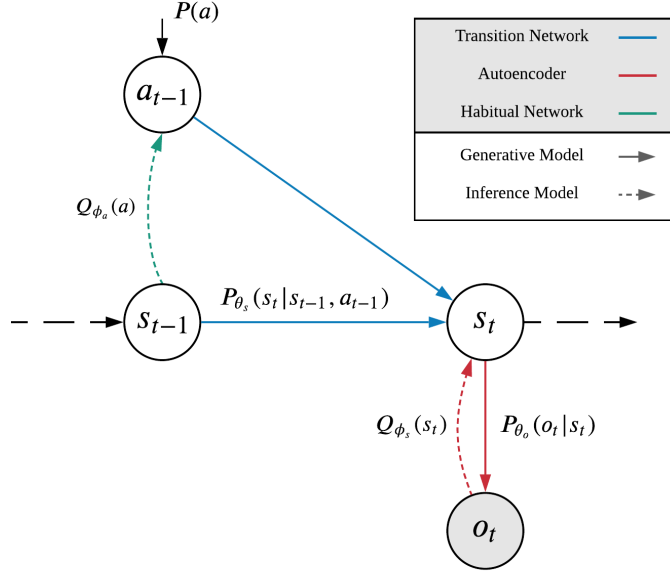


Figure 7: Generative model of the baseline agent.

Here, the main components of the generative model are: transition model $P_{\theta_s}(s_t | s_{t-1}, a_{t-1})$ parametrised by θ_s and later denoted as a function (neural network) that takes a state-action pair as input $f_{\theta_s}(\cdot, \cdot)$, and latent state decoder $P_{\theta_o}(o_t | s_t)$ parametrised by θ_o . The latent states, s , are assumed to be Gaussian-distributed with a diagonal covariance, where as observations, o , are Bernoulli-distributed. Note, that every component of the generative and inference models is parametrised by a neural network. The details of the implementation and training can be found in Appendix C.1.

Importantly, we modify the original transition model from Fountas et al. [10] to predict *the change in state*, rather than the full state⁴. Hence, the multivariate random variables are related by $\tilde{s}_t = s_{t-1} + \tilde{f}_{\theta_s}(s_{t-1}, a_{t-1})$, where $\tilde{f}_{\theta_s}(s_{t-1}, a_{t-1})$ is a Gaussian multivariate random variable with sufficient statistics given by $(\mu_{f_{\theta_s}}, \sigma_{f_{\theta_s}})$ predicted by the transition network and s_{t-1} is a vector representing latent state at time $t - 1$. Thus, the next state is also a Gaussian-distributed multivariate random variable \tilde{s}_t , with sufficient statistics:

$$\begin{aligned} \mu_{s_t} &= s_{t-1} + \mu_{f_{\theta_s}} \\ \sigma_{s_t} &= \sigma_{f_{\theta_s}} \end{aligned} \tag{18}$$

⁴This has largely become common practice in the field of model-based RL [17], improving algorithm efficiency and accuracy especially in environments with slow temporal dynamics (small change in state given an action).

Our model also includes two inference networks, which are trained using amortized inference⁵: habitual network $Q_{\phi_a}(a_t)$ parametrised by ϕ_a and observation encoder $Q_{\phi_s}(s_t)$ parametrised by ϕ_s . The habitual network acts as a model-free component of the system, learning to map inferred states directly to actions. As will be seen shortly, the fast habitual network will be used in conjunction with the slow MCTS planner.

We can now define the variational free energy for an arbitrary time-step t ; following Fountas et al. [10],

$$\begin{aligned} F_t = & -\mathbb{E}_{Q(s_t)} [\log P_{\theta_o}(o_t|s_t)] \\ & + D_{KL} [Q_{\phi_s}(s_t) || P_{\theta_s}(s_t|s_{t-1}, a_{t-1})] \\ & + \mathbb{E}_{Q(s_t)} [D_{KL} [Q_{\phi_a}(a_t) || P(a_t)]] \end{aligned} \quad (19)$$

where, importantly, $P(a) = \sum_{\pi: a_1=a} P(\pi)$ – summed probability of all policies beginning with action a (refer to Eq. 9). All the divergence terms are computable in closed-form, given our assumptions about Gaussian- and Bernoulli-distributed variables.

Finally, we define the EFE for our generative model. Here, we note that the neural network parameters of the generative model, $\theta = \{\theta_o, \theta_s\}$, are random variables and are, therefore, included in the joint probability distribution. Recalling equation 8, we again follow [10] and define EFE up to some time horizon T :

$$G(\pi) = \sum_{\tau=t}^T G(\pi, \tau) = \sum_{\tau=t}^T \mathbb{E}_{\tilde{Q}} [\log Q(s_\tau, \theta|\pi) - \log P(o_\tau, s_\tau, \theta|\pi)], \quad (20)$$

where

$$\begin{aligned} \tilde{Q} &= Q(o_\tau, s_\tau, \theta|\pi) \\ P(o_\tau, s_\tau, \theta|\pi) &= P(o_\tau|\pi)Q(s_\tau|o_\tau, \pi)P(\theta|s_\tau, o_\tau, \pi). \end{aligned} \quad (21)$$

As in the preliminaries, we can decompose Eq. 20 into its meaningful constituents,

$$\begin{aligned} G(\pi, \tau) = & -\mathbb{E}_{\tilde{Q}} [\log P(o_\tau|\pi)] \\ & + \mathbb{E}_{\tilde{Q}} [\log Q(s_\tau|\pi) - \log P(s_\tau|o_\tau, \pi)] \\ & + \mathbb{E}_{\tilde{Q}} [\log Q(\theta|s_\tau, \pi) - \log P(\theta|s_\tau, o_\tau, \pi)]. \end{aligned} \quad (22)$$

Here, we note that:

- $\mathbb{E}_{\tilde{Q}} [\log P(o_\tau|\pi)]$ is the likelihood of our prior observations under the generated observations, meaning that the expression corresponds to the notion of a reward.
- $\mathbb{E}_{\tilde{Q}} [\log Q(s_\tau|\pi) - \log P(s_\tau|o_\tau, \pi)]$ is the epistemic value of the hidden state, which intrinsically drives the agent to explore the environment.
- $\mathbb{E}_{\tilde{Q}} [\log Q(\theta|s_\tau, \pi) - \log P(\theta|s_\tau, o_\tau, \pi)]$ incorporates uncertainty about the agent’s model parameters, driving the agent to, similarly, resolve uncertainty about its model parameters.

⁵Exact training procedure will be discussed later.

To make expression 20 computationally feasible, we follow [10] decomposition,

$$\begin{aligned}
G(\pi, \tau) = & -\mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)Q(o_\tau|s_\tau,\theta,\pi)} [\log P(o_\tau|\pi)] \\
& + \mathbb{E}_{Q(\theta|\pi)} [\mathbb{E}_{Q(o_\tau|\theta,\pi)} H(s_\tau|o_\tau, \pi) - H(s_\tau|\pi)] \\
& + \mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)} [H(o_\tau|s_\tau, \theta, \pi)] \\
& - \mathbb{E}_{Q(s_\tau|\pi)} [H(o_\tau|s_\tau, \pi)].
\end{aligned} \tag{23}$$

Now, all the terms are easily computable. Specifically, expectations can be taken by performing sequential sampling of θ , s_τ and o_τ – essentially performing a roll-out by imagining future observations at each time τ . Note, that the prediction of the next state using the change in state is implicit in these mathematical formulations. Similarly, network parameters, θ , are sampled using MC dropout [40] and entropy terms can be calculated in closed-form given our assumption of Gaussian and Bernoulli random variables.

Authors of the paper also implement *top-down attention* mechanism by introducing variable ω , which is used to modulate uncertainty about hidden states. Specifically, the latent state distribution is defined as a Gaussian such that $s \sim N(\mu, \sigma^2/\omega)$, where μ and σ are calculated using Eq. 18, and ω is

$$\omega = \frac{a}{1 + e^{-\frac{D-b}{c}}} + d \tag{24}$$

where $D = D_{KL}[Q_{\phi_a}(a)||P(a)]$ and a, b, c, d are hyperparameters. Top-down attention is meant to promote latent state disentanglement and more efficient learning, as indicated by the empirical results in [10].

Finally, as explained in Section 2.3.2, action selection is aided with MC tree search, ensuring a more efficient trajectory search. Specifically, MCTS generates a weighted tree that is used to sample policies from the current time-step, where the weights refer to the agent’s estimation of the EFE given a state-action pair, $\tilde{G}(s, a)$. The nodes of the tree are predicted via the transition model, $P_{\theta_s}(s_t|s_{t-1}, a_{t-1})$ – see Figure 6: Planning. Using the standard MCTS formulation, the authors define the following upper confidence bound [111]:

$$U(s, a) = \tilde{G}(s, a) + c_{explore} \cdot Q(s) \cdot \frac{1}{N(s, a) + 1} \tag{25}$$

where $N(s, a)$ is the number of times action a has been taken from state s and $c_{explore}$ is a hyperparameter controlling the exploration of the tree. At any given state in the search tree, trajectory actions are sampled from $\sigma(U(s, a))$ if the state has been explored, and using the habitual prior $Q_{\phi_a}(a)$, if otherwise. At the end of the search, the MCTS is used to construct the action prior, $P(a) = \frac{N(a_i, s)}{\sum_j N(a_j, s)}$, from which the next trajectory of the agent, consisting of actions a_i , will be sampled. Furthermore, the planner is equipped with a threshold hyperparameter, T_{dec} , such that if the habitual network finds a clear option for the next action – it will be taken without continuing the search.

4 Prioritised Replay Buffer

The general question that this section of the project will attempt to answer is:

Given the baseline architecture outlined in Section 3, what data should our on-line agent train on?

At close inspection of the baseline architecture, we note three major problems. First, an on-line agent discards observations after one training iteration. This implies that observations that occur in the environment rarely (e.g. multiple spheres positioned close to each other) will also rarely occur in the training batches. In turn, this means that neural networks will tend to perform well on the most frequently occurring observations and poorly on rarely-occurring ones. In a testbed like Animal-AI, this can lead to a highly unfavourable behaviour, given that the majority of agents’ observations are likely to be walls or the overview of the environment from far. Moreover, even if the network learns some particular rarely-occurring transition, it is likely that, given enough time, this transition will be forgotten (via *catastrophic forgetting*) [78]. Second, an on-line agent will train on temporally correlated data, which is widely considered detrimental for training neural networks [78]. Third, an on-line agent’s only means of sampling new data is by taking actions in the environment; yet, this data may not be most optimal for the next training iteration – the agent is largely constrained by its current position in the environment to acquire training data.

Thus, we propose an introduction of a stochastic Prioritised Replay Buffer (PRB) [78] (also discussed in Section 2.4.2) which uses a weak form of episodic memory⁶. This component is aimed at tackling all three of the above issues by storing the transitions experienced by an agent in a buffer and later sampling batches of data for training, giving priority to transitions deemed more useful. Originally introduced by DeepMind, PRB (also named Prioritised Experience Replay) was used to improve training efficiency and effectiveness of a Deep Q-Learning agent by sampling transitions with a probability proportional to the produced temporal-difference error [78]. Similarly, in this project, we implement PRB to promote training on transitions that produce the highest values of variational free energy with respect to the transition model. We arrive at this by performing a qualitative comparison of different prioritisation strategies based on the different components of the total variational free energy (Eq. 5) in Section 4.2. Finally, we compare the qualitative and quantitative performances of on-line and PRB agents on a set of randomly generated environments in Section 4.3.

4.1 Details

As discussed, our agent possesses *three* main components – a transition network that predicts the next state, a variational autoencoder for encoding observations and decoding states, and a habitual network that directly maps states to actions. In turn, each of these components is

⁶By *weak*, we refer to the fact that memories stored in PRB will contain no references to other episodic memories. In turn, this implies that memories do not hold information about temporal dependencies between them. The *stronger* form of episodic memory will be used in Section 5.

trained on the corresponding loss function quantified by the variational free energy. Recall the total free energy (Eq. 5),

$$F_t = -\mathbb{E}_{Q(s_t)} [\log P_{\theta_o}(o_t|s_t)] \quad (26a)$$

$$+ D_{KL} [Q_{\phi_s}(s_t) || P_{\theta_s}(s_t|s_{t-1}, a_{t-1})] \quad (26b)$$

$$+ \mathbb{E}_{Q(s_t)} [D_{KL} [Q_{\phi_a}(a_t) || P(a_t)]] \quad (26c)$$

Here, expressions (26a) and (26b) are used to train the variational autoencoder, expression (26b) is used for the transition network, and expression (26c) is used for the habitual network. The components of the total free energy correspond to a measure of belief update for each of the networks, and therefore, loosely speaking, quantify the prediction error generated by each of the networks. We employ these quantities in the implementation of PRB by sampling training data with a probability proportional to the values of the free energies.

As mentioned, PRB stores *all*⁷ transitions experienced by the agent in an environment. A single transition is represented as a tuple (o_t, a_t, o_{t+1}) , where o_t and a_t are an observation and action taken at time t , and o_{t+1} is an observation that followed at the next time-step. The probability of a single transition to be sampled from the buffer is calculated by

$$p_n(i) = \frac{F_{n,i}^\alpha}{\sum_k F_{n,k}^\alpha}, \quad (27)$$

where $n \in \{ \text{transition}, \text{autoencoder}, \text{habitual} \}$ referring to a network to be trained, $p_n(i)$ is the probability of the i -th transition to be sampled for training network n , $F_{n,i}$ is the variational free energy associated with the i -th transition and network n , and α exponent is a hyperparameter used to control the degree of prioritisation [78]. In the final version of the system, we fix $n = \text{transition}$, as prioritisation with respect to the transition model resulted in the best training samples. This will be discussed in the next section.

Furthermore, changing the sampling distribution in such a manner introduces significant bias to the stochastic gradient updates. It is, thus, necessary to correct this bias by introducing importance sampling weight term [78], such that

$$w_{n,i} = \left(\frac{1}{N} \cdot \frac{1}{p_n(i)} \right)^\beta, \quad (28)$$

where $w_{n,i}$ is the importance sampling weight for the i -th transition in the buffer and for training network n , N is the number of sampled transitions, and β is a hyperparameter used for annealing the importance sampling weight. At training, the loss of a corresponding network is multiplied by this weight, $w_{n,i} \cdot F_{n,i}^\alpha$, thus realising weighted importance sampling. Moreover, β is linearly increased during the entire training procedure until the maximum value of 1, as suggested in Schaul et al. [78].

Given the large size of the buffer (100,000 transitions), calculating the probabilities and performing sampling according to these probabilities can be computationally costly. To

⁷Up to the size of the buffer

tackle these challenges, we update the free energy values only for the transitions sampled in the latest batch (instead of updating every single transition value in the buffer after every training iteration) and implement an efficient sampling data structure called ‘sum-tree’ [78], allowing updating and sampling in $O(\log N)$. See Appendix A for implementation details.

4.2 Priority Metric

In this section, we perform a *qualitative* comparison of the different metrics, proportional to which the data can be sampled from the buffer. Due to the time and computational constraints of this project, we are unable to compare the quantitative performance differences; hence, we conduct a thorough inspection of the observations stored in a buffer during the learning procedure by sorting observations in the descending order by:

1. Habitual network’s, Q_{ϕ_a} , variational free energies, Eq. 26c;
2. Transition model’s, P_{θ_s} , variational free energies, Eq. 26b;
3. Autoencoder’s, Q_{ϕ_s} and P_{θ_o} , variational free energies, Eqs. 26a + 26b;

Figure 8 shows samples from the buffer after $\sim 250k$ training iterations. Each column shows samples sorted (in a descending order) by the corresponding values of free energy produced by each of the networks – habitual network, transition model, and autoencoder. Importantly, the trends illustrated in this figure are not specific to the chosen range, but rather show a more general pattern. To analyse these figures, we pay special attention to samples bounded with red colour, since these represent training observations that will be prioritised during training.

By inspecting the observations, it is immediately evident that the samples with respect to the transition model constitute **more favourable states** with respect to the potential actions an agent can take. This is in line with the formulation of the transition-model variational free energy (Eq. 26b) that is intrinsically linked to action. In general, these samples are characterised by: i) the strong presence of objects in different configurations, ii) objects that only recently appeared in the frame (e.g. turning and seeing a green sphere), and iii) yellow spheres that disappear at the next frame⁸. Here, we address two important questions:

Why are these states favourable? To answer this, we must recall that an active inference agent aims to minimise the variational free energy. It does so by imagining possible futures from the current state and choosing an action that corresponded to the futures with the lowest value of the expected free energy. Therefore, the shown states are favourable because they provide the agent with means of minimising the free energy, i.e. states with the highest number of affordances with respect to the objects in the environment. Because objects are a central component of the environment and *the only means of minimising the free energy*

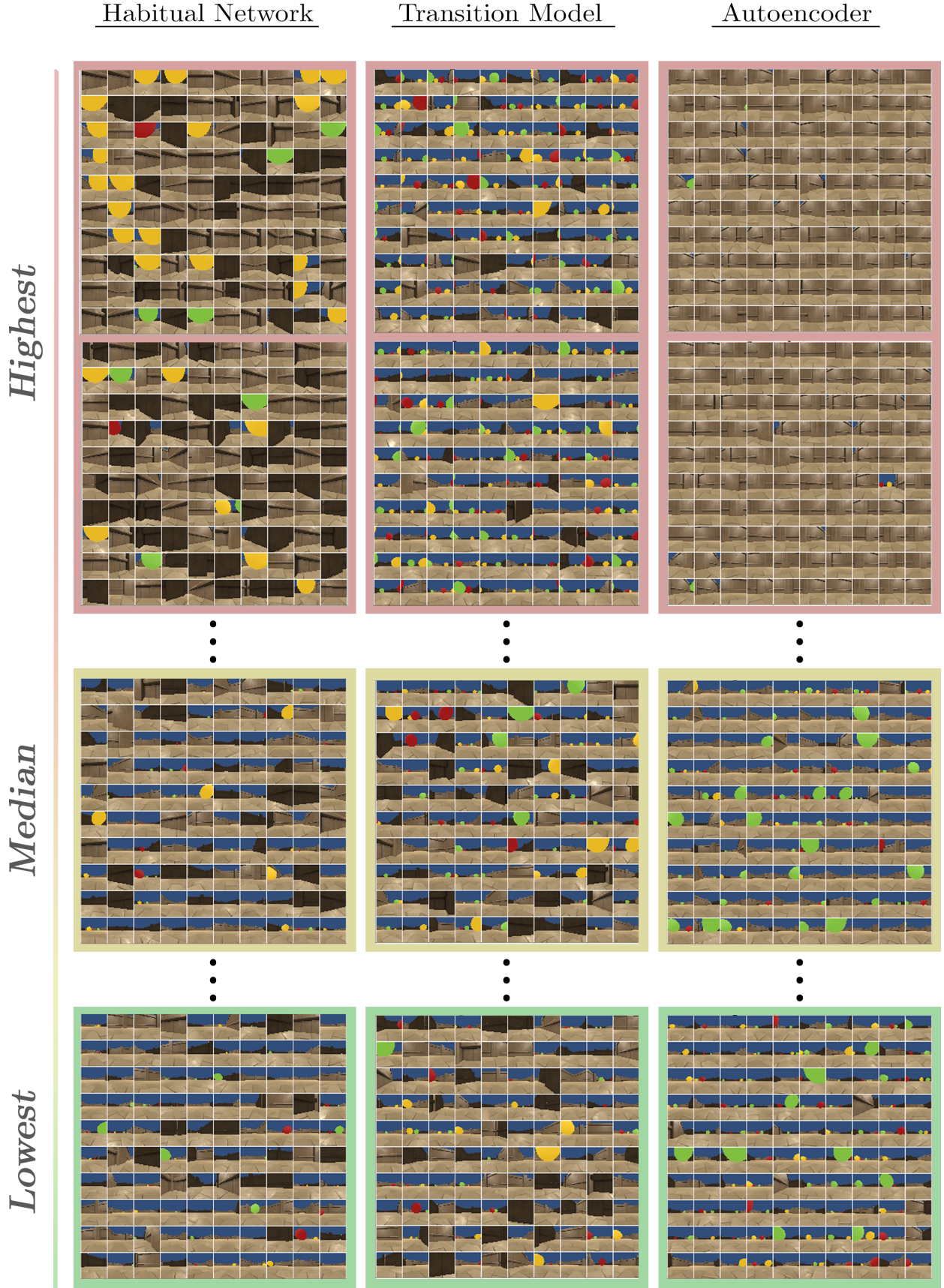
⁸Samples shown in Figure 8 constitute the *first* frames from the transitions, which normally include a pair of observations (o_0, o_1). This was done to exemplify the presence of the disappearing yellow spheres in the buffer, which are considered *surprising* events for the transition model.

(recall the observational prior, Figure 5), it is most important for an agent’s generative model to learn these representations in highest detail.

Why are these states characterised by the presence of objects? We can categorise the transitions that cause high transition-model prediction error into two main groups: epistemic surprise and model-imperfection surprise. The former refers to transitions that the model could not have predicted accurately due to the lack of information – disappearing spheres or spheres that appear in the frame upon turning. Although these events do appear at the top of the buffer, they do not constitute the main bulk of ‘surprising’ observations. Rather, the main bulk of these high prediction-error transitions fall into the latter group. It is useful to think of these observations as being rich in *combinatorial structure*. Because the transition model cannot be assumed to be perfect, nor can it be considered to generalise perfectly, one of the major sources of transition model prediction error would be the different contexts in which the objects would appear in the frame. Their relative positions, their size, their colours, their background are all contributing components to imperfect predictions of the transition model. In turn, the effects of these components compound in the presence of objects, which explains their abundant presence in high-error samples.

In line with the above explanations, we note that the least surprising states of the transition model largely involve far-away scenery and walls. Importantly, these states are not directly related to the exploitative component of an active inference agent. That is, these observations do not provide an agent with immediate information to plan its path to the goal, which means an agent would engage in exploring the environment instead.

In contrast to the transition model, samples with respect to the highest free energies of the autoencoder and the habitual network mostly involve walls. As discussed these states are not necessary to be learned with great detail and would, therefore, be a wasteful way to train the generative model. Additionally, the habitual network samples with high free energies often contain close green or yellow spheres. This can be explained by the generative model’s inability to produce consistent predictions of what happens after the sphere is retrieved. As a result, the habitual network is not able to mimic the behaviour of the planner in such states, thus producing large prediction errors.

Figure 8: Buffer observations *sorted by free energy* values in the descending order.

We conclude that prioritisation based on the free energy values of the transition model are most favourable for training a generative model, whose primary aim is to simulate agent’s future observations with respect to its actions and the objects in the environment. Interestingly, it is counter-intuitive to discard the idea of training each individual network with prioritisation based on their individual prediction errors. After all, a neural network should, theoretically, get better at predicting those samples as the training goes on, thus learning faster. However, we stress that learning a perfect generative model is not necessary for solving the task of retrieving a green sphere (e.g. perfectly modeling the environment’s floor tiles). In this project, we take that learning the generative model with respect to states with affordances useful for the minimisation of the variational free energy is most effective. This idea was also discussed in the literature review, where a number of papers proposed that in order to solve tasks in an environment, the generative model does not need to be *perfect*, but rather it must be intrinsically linked to action and agent’s affordances [6, 74, 73]. Furthermore, empirical results show that the presence of walls as samples with the highest free energies (of habitual network and autoencoder) persist over the entire course of training. This is likely an interesting side-effect of the Animal-AI environment, where walls and the floor tiles contain greater graphical detail than the rest of the environment. Because of this, the autoencoder struggles to map wall observations onto the latent state, as well as to reconstruct them with high fidelity. As a result of the bottleneck, which is the autoencoder’s latent dimension, it will never learn to do it – and so the high values of free energy persist. This also explains the presence of walls in the habitual network that tries to map the poorly-represented states to actions.

Additionally, the use of the *total* variational free energy as the priority metric was discarded based on the above analysis of its individual constituents. Nevertheless, the definition of alternative metrics remains an open direction for future research (Section 6).

In summary, we reconcile the use of PRB with the framework of active inference. Originally used to improve the efficiency and performance of a Deep Q-network [78], the system prioritised samples that produced higher temporal-difference errors. In contrast, we use prioritisation to improve the quality of the agent’s generative model with respect to transitions that resulted in higher values of transition-model free energies.

4.3 Experimental Results

Figure 9 shows a qualitative comparison of the generative models of an on-line agent vs. an agent with a replay buffer employing prioritisation according to transition free energies. Both models have been trained for 750k steps – more details can be found in Appendix C.1. The displayed imagination roll-outs were hand-picked to display some of the most important features for the comparison. In practice, the quality of produced roll-outs will not be strikingly different on average; however, these differences become apparent in more rare situations. Importantly, this can make all the difference for the quantitative performance of an agent, as will be shown in the next pages.

It is evident that the generative model with PRB learns better object representations

and is able to better preserve them over time in a roll-out. In particular, the roll-outs of the on-line agents are characterized by disappearing spheres, showing what would be the average observation in the environment instead. On the other hand, PRB agent much better preserves information about the spheres, which would be crucial for making the right decisions in an episode. More results showing improved performance of the autoencoder can be found in Appendix B.

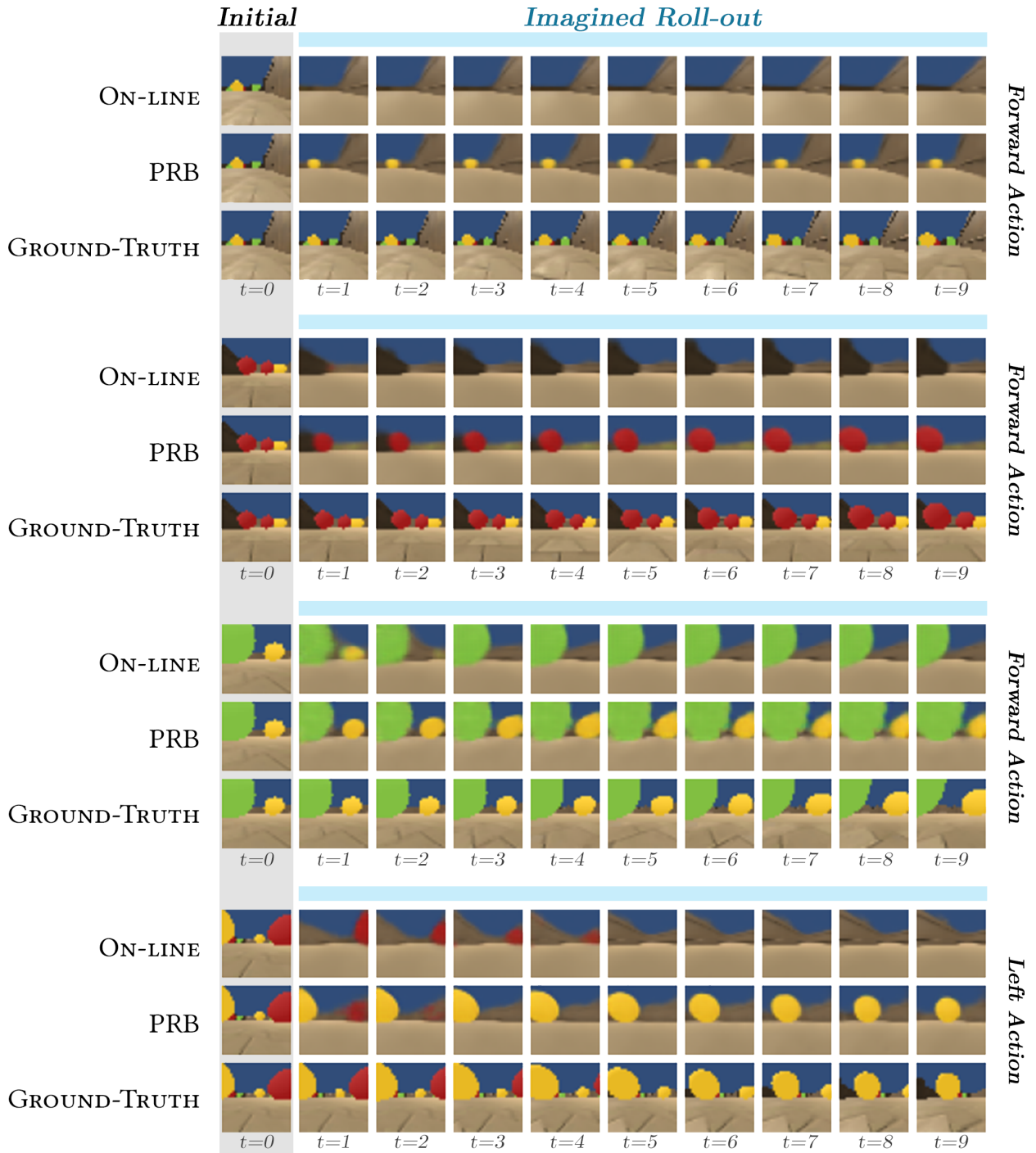


Figure 9: Comparison of on-line vs. PRB imagination roll-outs.

Furthermore, we test the agents on a set of randomly generated environments, containing one green sphere (terminal positive reward), one yellow sphere (positive reward), and one red sphere (terminal negative reward). Agents are restricted to the total of 100,000 steps, after which the testing terminates. Importantly, these configurations are characterised by sparse rewards and relatively little episode time (max 500 steps before the environment is terminated). The agent must, therefore, quickly find the reward by exploring its neighbourhood, whilst avoiding the negative reward. Here, the quality of the learned generative and inference models is of great importance for two main reasons:

1. The agent’s transition model is perfectly myopic, because it is a simple feed-forward neural network. This means that an agent’s current frame of observation is *the only* information it has to reason about the next action. If the agent has seen a faint silhouette of a green sphere on the side, yet made a turn to drive it out of the frame, it has no means of remembering that the sphere was there. In an environment with sparse rewards, one such algorithm decision would significantly reduce the agent’s chances of getting a reward in an episode. It is for this reason that a generative model that can better deal with object representations and more accurately perform imagination roll-outs with respect to objects will be in advantage.
2. Better object-centric representations in the latent state play an important role for the agent’s ability to see objects far away. If the agent’s inference model does not encode spheres that are small (due to the signal being too small) into its latent representation, the generative model will not predict its presence in the next states. This is largely illustrated in the on-line agent’s first two imagination roll-outs, Figure 9. Thus, an environment with sparse rewards that often appear far away from an agent would put its generative model to test.

Furthermore, the MCTS *was not* used to test the agents in this section, given extensive problems related to the system’s behaviour under it. However, the problems related to MCTS are not relevant for testing the quality of an agent’s generative model with and without the PRB. Although these will be discussed in Section 5, we would like to briefly mention them here.

In short, the use of MCTS results in agents getting stuck at walls, which significantly increases the length of testing, as the general trends of the agent’s progress cannot be tracked reasonably well. The reason for why MCTS-based agent gets stuck at walls is actually a major motivation for the second part of the report. Recall that MCTS works by sampling different trajectories according to the pre-defined upper-bound and the habitual prior $Q_{\phi_s}(a)$. After the potential trajectories are defined, an agent executes the actions of the entire trajectory with the lowest associated EFE. Thus, re-planning of its next actions occurs only after the agent finishes executing the chosen sequence of actions. At the same time, as we will see in Section 5.1 (Scenario 1), the agent’s transition dynamics model is slow and does not predict far into the future. For an agent at a wall, this means that its imagination roll-outs are largely populated with images of walls, deeming every action as almost identical. In the process of MCTS planning, this would result in trajectories that

keep the the agent at a wall for a long time, since to leave it, an agent would have to take a number of *consecutive* left or right actions. However, instead, MCTS are likely to include rather diverse trajectories (e.g. right, right, forward, left, forward, etc.). Because of this trajectory flexibility and relatively infrequent re-planning under MCTS, the agent spends much more time at the walls. These reasons will become more apparent in Section 5.

Therefore, instead, we perform a form of model predictive control (MPC), in which an agent calculates its next action *after every step*. Furthermore, an agent evaluates only *three* policies – N times left, N times forward, and N times right, where N is the number of times an action is repeated and was chosen to be 5. The agent would then roll-out imagined observations (using its generative model) for the three given policies, calculate the EFE for each policy and take an action corresponding to the lowest calculated EFE. Although this method is more computationally expensive than MCTS, it allows us to better study the performance of on-line and PRB agents.

Quantitative results presented in Figures 10 and 11 support the initial hypothesis. It can be seen that the PRB agent shows significant improvement, retrieving rewards faster, while also being better at avoiding terminal red rewards.

More generally, the results also show that the active inference agents do not only learn to retrieve green rewards (which match our observational prior from Figure 5), but they are also attracted to yellow rewards. This can be explained by the dominating exploratory component of the EFE, which encourages the agents to explore states with high uncertainty. Because yellow spheres disappear at retrieval, the agent’s generative model has high uncertainty about the states that come after (i.e. what is behind the sphere?). However, conveniently, the colour of yellow spheres is also relatively close to green in the pixel space, which provides additional incentive for an agent to reach them. On the other hand, red spheres do not constitute means of minimising free energy with respect to the green observational prior. As can be seen, these tend to be avoided.

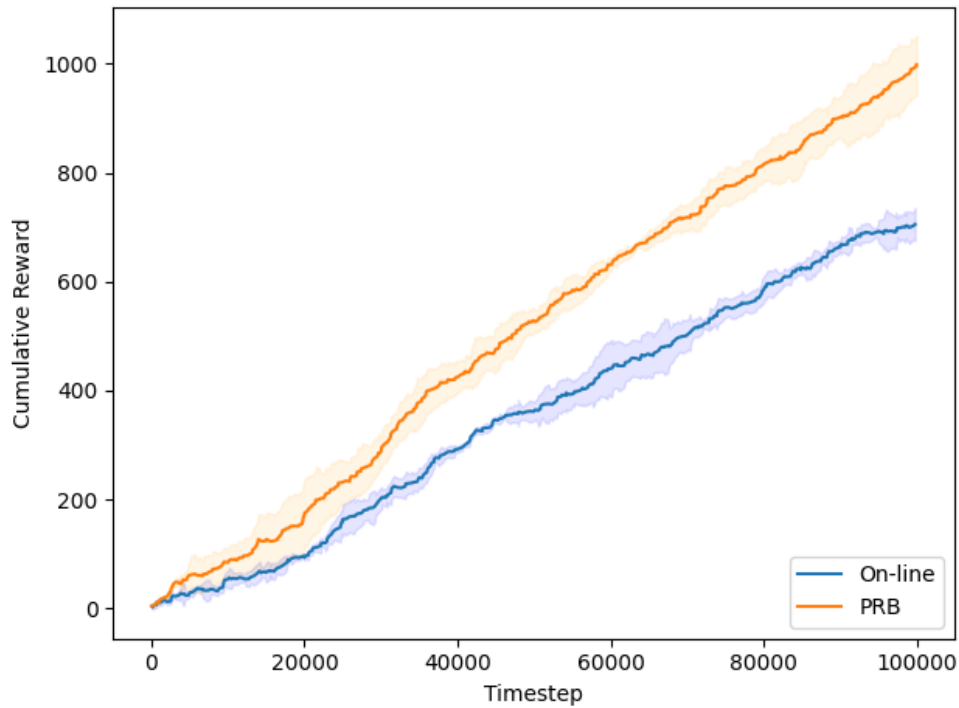


Figure 10: **Experimental results.** Comparison of mean cumulative rewards after 100,000 steps. Shaded regions represent two standard deviations (over 5 runs) from the mean lines.

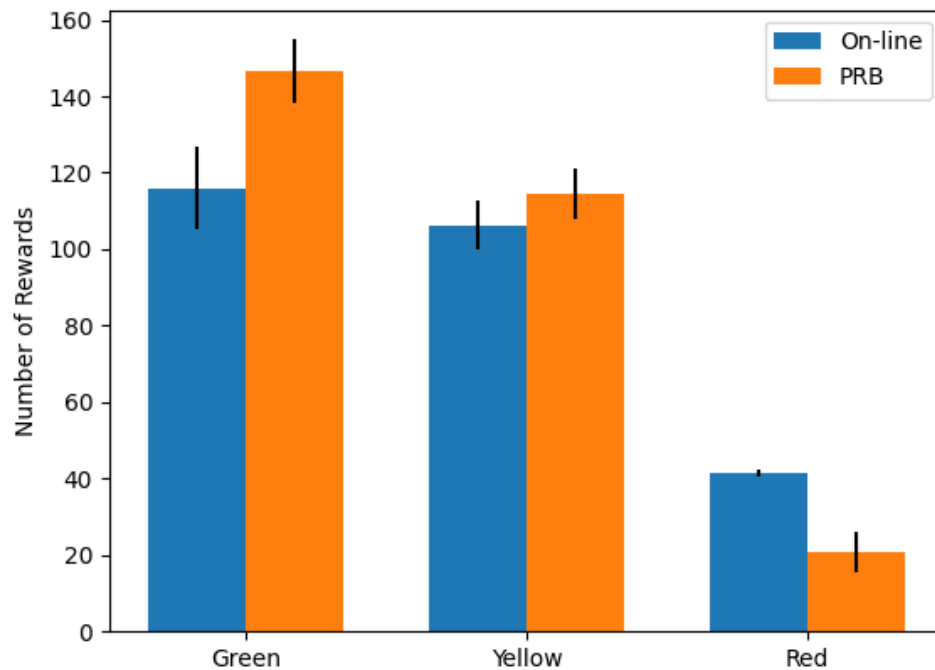


Figure 11: **Mean number of collected rewards by category.** PRB agent systematically retrieves more green and yellow rewards, while better avoiding negative red rewards. Black lines represent two standard deviations (over 5 runs) from the mean.

5 Temporal Episodic Memory Module

Temporal Episodic Memory module (TEM) constitutes the second large part of this project. In contrast to PRB, here we employ a *stronger* form of episodic memory that additionally incorporates information about temporal dependencies between memories. TEM was largely inspired by three things:

- Problems with model-bias and sequential error accumulation in model-based reinforcement learning; although there is a number of works attempting to tackle these issues directly, most develop mitigating techniques with the use of uncertainty quantification (Section 2.2.3).
- Neuroscience literature that studies the functions of episodic memory in the context of assisting humans in *imagination* (constructing novel future scenarios) and *mental time-travel* (Section 2.4.2).
- Recent works on defining (learning) a subjective timescale with the use of perceptual content of sensory inputs (Section 2.4.2).

As such, this part of the project will address two questions at once:

1. Can we come up with a principled way to learn the transition dynamics model over a more useful, subjective timescale of the environment?
2. Can we learn a transition dynamics model that is capable of imagining ‘novel’ observations in the future?

Because these two objectives are quite vague in terms of terminology, we first want to provide two motivating examples that would justify the objectives and explain the ambiguous terms.

5.1 Motivating Examples

Scenario 1. Suppose an agent observes a green sphere in the distance as shown in Figure 12. Which way will the baseline agent go? To answer this, we must first and foremost consider the main incentive of an active inference agent – which is to minimise the variational free energy. One of the central components of the free energy is the observational prior, which is the observation that the agent is trying to reach by taking actions in an environment. Incidentally, the bright walls of the environment provide one of the ways to minimise the free energy *sub-optimally* with respect to the green observational prior. This is because the light-brown colour of the walls is relatively close to the green colour of a sphere in the pixel space. Recall that our agent will first perform imagination roll-outs in different directions, imagining the possible observations it may encounter in the near future. Next, it will calculate the expected free energy for each of the roll-outs and sample an action with probability proportional to the negative expected free energy. However, given that the ground-truth, objective dynamics of the environment is *slow*, the green sphere will only marginally increase in size, as the agent moves forward. On the other hand, by turning to the nearest wall, the agent can much faster minimise the variational free energy. Therefore,

even though the green sphere represents the optimal state for the minimisation of the free energy, the agent simply *has no means to know* that going forward will get it to the optimal state, given the slow timescale of its transition model. If, however, the agent was equipped with a transition model that performs temporal jumps and predicts a much faster approach to the green sphere, it would choose to go forward instead. This motivates the learning of a transition model over more useful, subjective timescales.

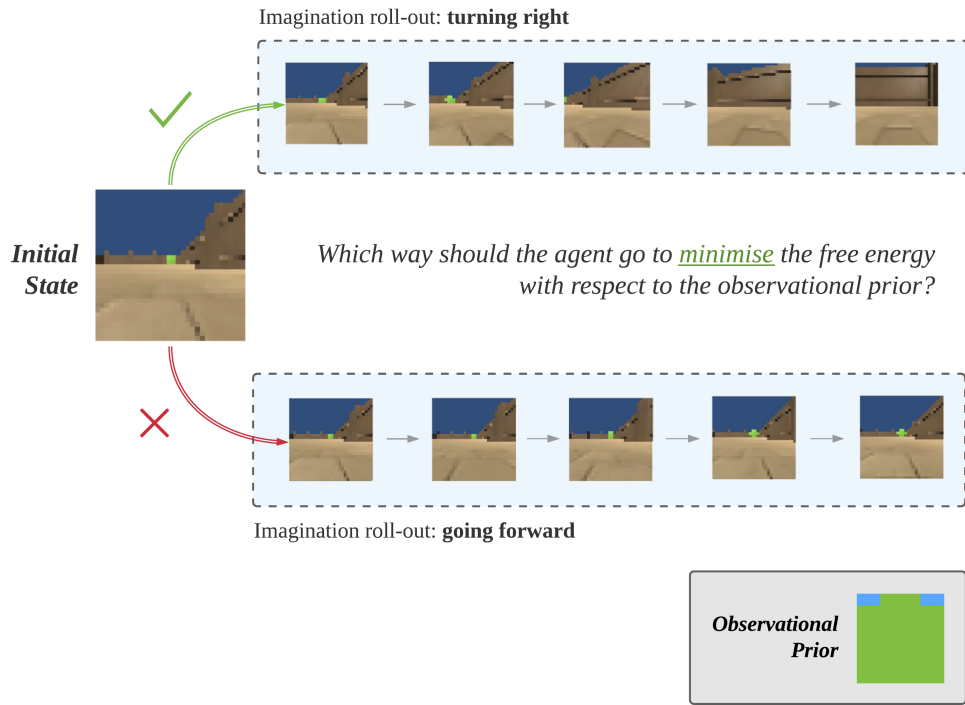


Figure 12: **Motivating example 1.** In the scenario, where a green or yellow sphere is far away, slow timescale will incentivise the agent to choose to go to a wall, which minimises the free energy in a sub-optimal way.

Scenario 2. Suppose an agent reaches a light-brown wall. Its colour provides the agent with a sub-optimal way to minimise the free energy. At the same time, the agent is equipped with a transition model that learned the dynamics of walls fairly well, thus reducing the quantity of the exploration component of the EFE (Eq. 20). Performing the roll-outs using the slow transition dynamics model yields similar observations of walls, providing the agent with no incentive to leave its current state. In order for the agent to leave the wall, its imagination must provide it with states that lead to lower free energies. Let's, however, inspect what the slow transition dynamics would yield in Figure 13. We note two major things that would discourage the agent from leaving the wall. First, the slow timescales of the transition model would populate the imagination roll-outs with wall observations. Second, the slow transition dynamics model is trained to predict the most likely next state given the current state and action. In turn, this implies that the model would predict the *average* expected observation, which it observed during training in similar states. Given that the environment is sparsely populated with objects, the generative model predicts future states in absence of any objects, as can be observed in the top imagination roll-out

in Figure 13. These two problems are *corollaries of how the transition model is trained*. In particular, it is trained using the slow, ground-truth dynamics of the environment *and* on all experienced transitions. On the other hand, if the transition model was trained to predict further into the future (in line with Scenario 1) *and* predict states deemed as interesting by the agent (according to some metric), imagination roll-outs for turning right or left from the wall would contain states that would encourage the agent to leave and explore. Thus, in addition to the more long-term state predictions, the transition model should be capable of imagining interesting⁹ states, which, as we earlier argued in Section 4, are likely to contain objects that provide means for minimising the free energy. In that sense, this scenario describes the same problem as Scenario 1, where an agent has no means to know that, by leaving the wall, it could observe an optimal state for the minimisation of the free energy.

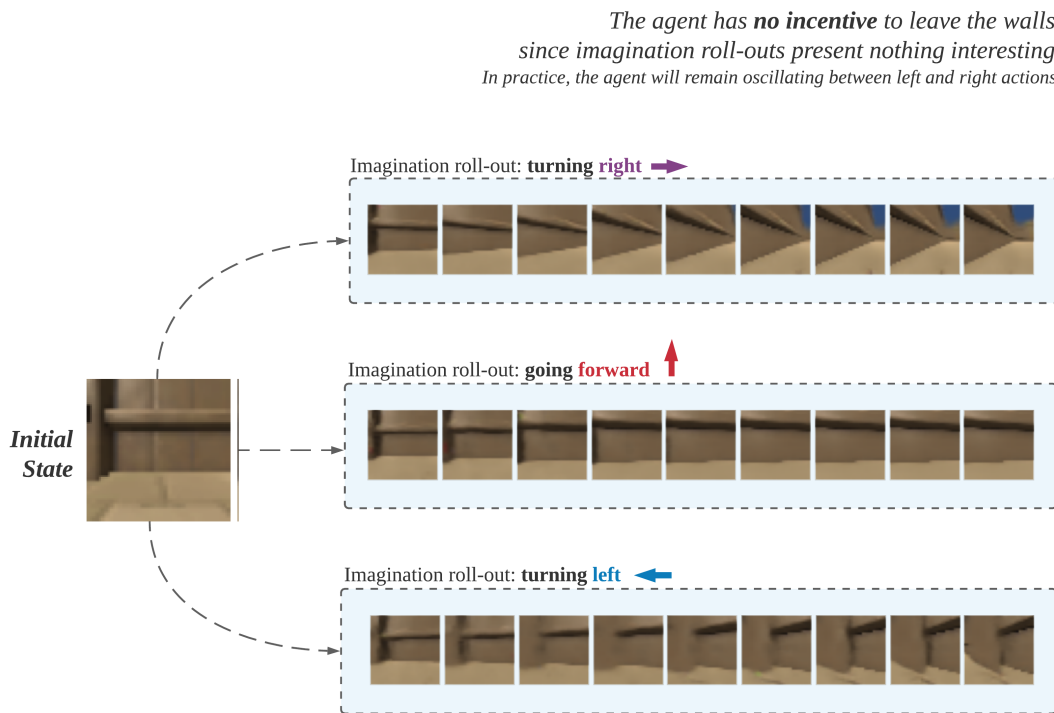


Figure 13: **Motivating example 2.** An agent equipped with a transition dynamics model that is both *slow* and *cannot imagine objects* will be discouraged to leave the wall, as no better means of minimising the free energy are provided in its imagination roll-outs.

Given the two motivating examples, we formulate the following characteristics of the alternative transition model:

1. **It must learn the dynamics of the environment over the *subjective timescale*,** which would be more useful for the agent’s action selection (see Scenario 1). As such, the subjective timescale is of variable rate, meaning that the rate of change of states should vary depending on the context the agent finds itself in. For instance, the subjective rate of time should be fast in the absence of anything interesting (thus

⁹The exact formulation of what constitutes an interesting state was foreshadowed in the Section 4 on PRB, and will be further explained in the coming pages.

predicting farther into the future) and should decrease as the agent approaches interesting states. An example of this is shown in Figure 14. This property of perceiving time is also characteristic to humans [105, 112, 106] and has been extensively studied in neuroscience.

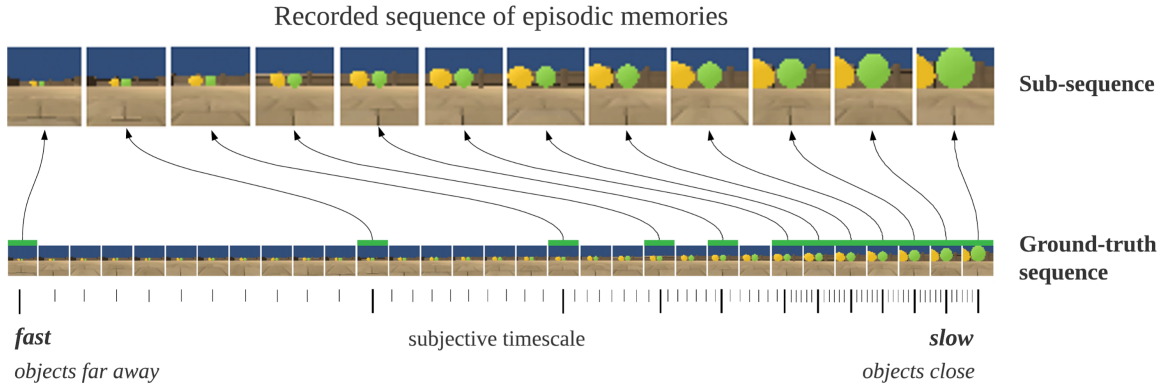


Figure 14: **Subjective timescale** of variable rate. Bottom sequence shows the ground-truth sequence of observations. Top (Sub-)sequence shows the formed episodic memories that define the subjective timescale.

2. **It must be capable of imagining objects**, where there may be none. In particular, if we inspect the original, slow transition model, we will find that the model predicts only what it observes. If the current observation contains no information about any objects in the surroundings, the original model will not be able to imagine any. This has two implications. First, it deems any action as almost identical in the absence of objects – since the transition model would simply predict the walls, the floor and the sky, given that they are the most likely (average) observations in the environment. Second, the transition model does not fully incorporate information contained in the training data. Specifically, the information contained in the training data is useful not only for learning the *physics* of the environment – i.e. given the current observation and the objects within it, how will the chosen action change the observation and the relative positions of the objects in the frame – but it is also useful for learning *temporal dependencies* between states. This means asking questions like: given the current state, what interesting states does the agent expect to see by turning right? Learning such dependencies between states can be crucial for an active inference agent that largely relies on its generative model of observations to make decisions in the environment. Importantly, this question cannot be answered if the transition model is expected to predict *the most likely* next states; therefore, we must come up with an alternative approach to train a transition model.

Hence, in what follows, we define one potential approach to learning a transition model with the afore-mentioned characteristics. We then perform extensive analysis of the imagination roll-outs produced by the system, and finally test it against the PRB agent from Section 4.

5.2 Architecture

The system described above by definition implies training only on selected observations. These selected observations define a sub-sequence of states and the subjective timescale, over which the TEM transition model is trained – see Figure 14. To implement the system, we create a class *EpisodicBank*, which contains two basic components – memory accumulation system and transition model trainer. To proceed with the explanations, we define a few important terms used hereinafter:

- A *ground-truth sequence* is a sequence of *all* states experienced in the environment during a single episode, $S_g = \{s_0, s_1, s_2, \dots, s_T\}$ – Figure 14: bottom row sequence.
- *Sub-sequence* (subjective sequence) is a sequence of states selectively picked by our system, and over which the new transition model would be learned, $S_e = \{s_{\tau_1}, s_{\tau_2}, s_{\tau_3}, \dots, s_{\tau_N}\}$ – Figure 14: top row sequence.
- Each observation in a Sub-sequence is called an *episodic memory* and consists of a set of sufficient statistics, $s = \{\mu_s, \sigma_s\}$, where μ_s and σ_s are mean and variance vectors of a Gaussian-distributed state s , respectively. Additionally, each episodic memory contains a reference to its preceding (parent) episodic memory and all actions taken after its formation. The process of recording Sub-sequences is called *memory accumulation*.
- *TEM transition model* is the dynamics model learned over the Sub-sequences, where as *transition model* is used to describe the dynamics model over the ground-truth sequences.

Memory Accumulation. An active inference agent is sent interacting with the environment under a pre-trained generative model with the use of PRB described in Section 4. During this process, each transition is evaluated based on the transition model free energy. As we saw in Section 4.2, the free energy with respect to the objective-timescale transition model, $D_{KL}[Q_{\phi_s}(s_t)||P_{\theta_s}(s_t|s_{t-1}, a_{t-1})]$, represents the degree of surprise¹⁰ experienced by the transition model upon taking an action. This can similarly be interpreted as the prediction error of the transition model. Inspired by the works from neuroscience on modeling human time perception [105, 106] and the recent work on video frame prediction [110], we take this metric as the criterion for forming a memory and, thus, defining the subjective timescale. As the agent moves in the environment, each transition is evaluated according to this metric; if the value of the free energy exceeds a pre-defined threshold, a memory is formed and placed into a Sub-sequence. At the end of each episode, the recorded Sub-sequence is sent to *EpisodicBank* and stored for later use.

¹⁰Note, it may be erroneous to consider the recorded events as surprising, though some of them certainly are. Rather, as mentioned in the PRB section, the main bulk of these events are events rich in *combinatorial structure*. Because the transition model cannot be assumed to be perfect, the major sources of transition model prediction error would be the different contexts in which objects would appear in the frame. Their relative positions, their size, their colours, their background are all contributing components to imperfect predictions of the transition model. As such, the effects of these components increase in the presence of objects, which is exactly the property we want the accumulation system to have.

TEM transition model. It is important for the TEM transition model to be an autoregressive model. As discussed, the purpose of the system is not only to learn the basic physics of the environment, but also to learn the temporal dependencies between the recorded states. Therefore, we train an LSTM [44] on the recorded Sub-sequences. We do this by randomly sampling a batch of 15 Sub-sequences (which may be of variable size), performing zero-padding to equalise their length to 50, and sending them for training. In our architecture, an LSTM calculates the hidden state h_τ at subjective time τ using a deterministic mapping,

$$h_\tau = f_{\theta_h}(s_\tau, a_\tau, h_{\tau-1}) = \sigma(x_\tau W_h + h_{\tau-1} U_h + b_h), \quad (29)$$

where s_τ and a_τ are the latent state and action taken at subjective time τ respectively, x_τ is the concatenated vector of s_τ and a_τ , and $\theta_h = \{W_h, U_h, b_h\}$ are deterministic LSTM model parameters. Importantly, function f_{θ_h} is deterministic and serves only to encode information about preceding steps into the hidden state of the LSTM. This hidden state h_τ is then mapped to a latent state $s_{\tau+1}$ at the next subjective time $\tau + 1$ via a feed-forward neural network with random-variable parameters, θ_{hs} , using $P_{\theta_{hs}}(s_{\tau+1}|h_\tau)$ with MC dropout. The parameters of both of the networks are trained via backpropagation with a loss function defined as

$$\begin{aligned} \mathcal{L} &= \sum_{\tau}^T D_{KL} \left[Q_{\phi_s}(s_{\tau+1}) || P_{\theta_{hs}}(s_{\tau+1}|h_\tau) \right] \\ &= \sum_{\tau}^T D_{KL} \left[Q_{\phi_s}(s_{\tau+1}) || P_{\theta_{hs}}(s_{\tau+1}|f_{\theta_h}(s_\tau, a_\tau, h_{\tau-1})) \right] \end{aligned} \quad (30)$$

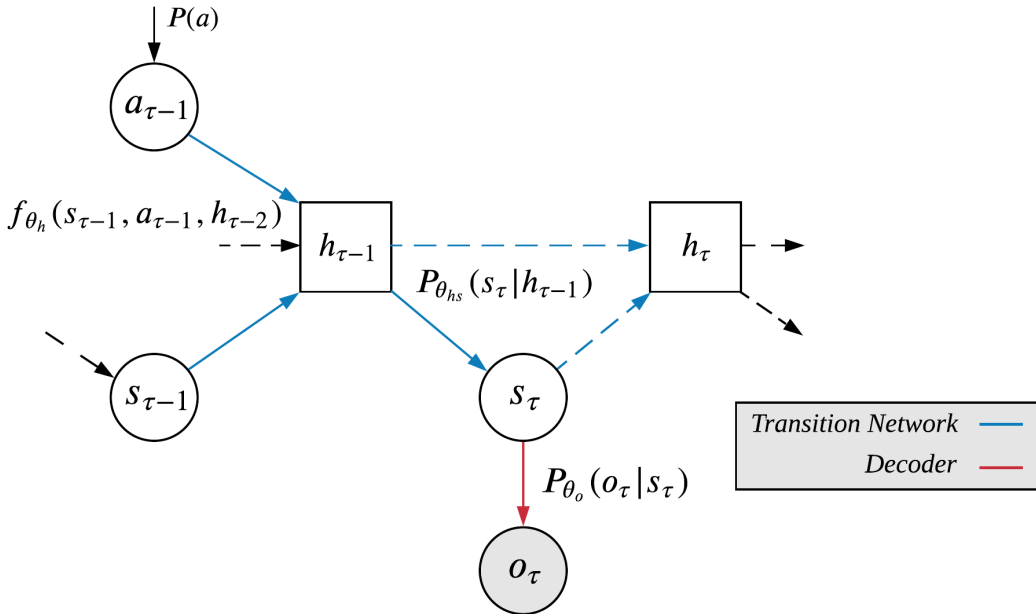


Figure 15: **The TEM generative model.** Graphical model of how observations are generated during imagination roll-out. Boxes and circles are deterministic and random variables, respectively.

The new generative model of observations is shown in Figure 15. Because the mapping of LSTM is deterministic, the formulation of the variational free energy remains intact with the exception of the second term that now includes the state prediction produced by the network $P_{\theta_{hs}}$ conditioned on the hidden state of the LSTM,

$$\begin{aligned}
 F_\tau = & -\mathbb{E}_{Q(s_\tau)} \left[\log P_{\theta_o}(o_\tau | s_\tau) \right] \\
 & + D_{KL} \left[Q_{\phi_s}(s_\tau) || P_{\theta_{hs}}(s_\tau | h_{\tau-1}) \right] \\
 & + \mathbb{E}_{Q(s_\tau)} \left[D_{KL} [Q_{\phi_a}(a_\tau) || P(a_\tau)] \right]
 \end{aligned} \tag{31}$$

Memory Accumulation and TEM Training

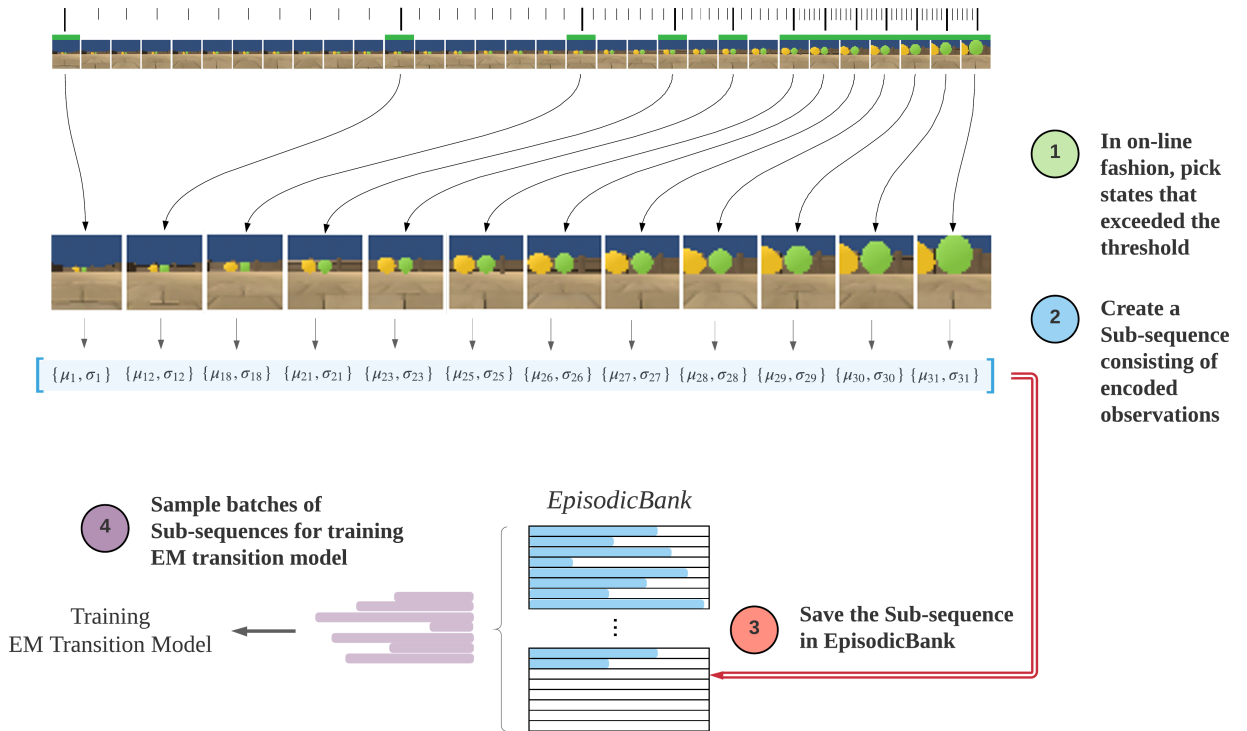


Figure 16: **The entire pipeline of training a TEM module.** First, episodic memories are accumulated according to the transition model free energy. Next, the Sub-sequences are formed as sequences of episodic memories, where each sequence refers to the memories recorded in a single episode. Lastly, Sub-sequences are sampled from EpisodicBank for training the TEM transition model.

Action Encoding. The initial training scheme of the TEM transition model included training on Sub-sequences in the form of $\{s_{\tau_1}, a_{\tau_1}, s_{\tau_2}, a_{\tau_2}, \dots\}$, where a_{τ_1} is the action taken at time τ_1 (i.e. the first action taken after memory s_{τ_1}). However, this system struggled to learn meaningful dependencies between actions and next states, see Figure 17. Specifically,

using a single action – that was made at the time-step of the preceding memory – to represent *an entire trajectory* (consisting of an arbitrary number of actions) is not ideal.

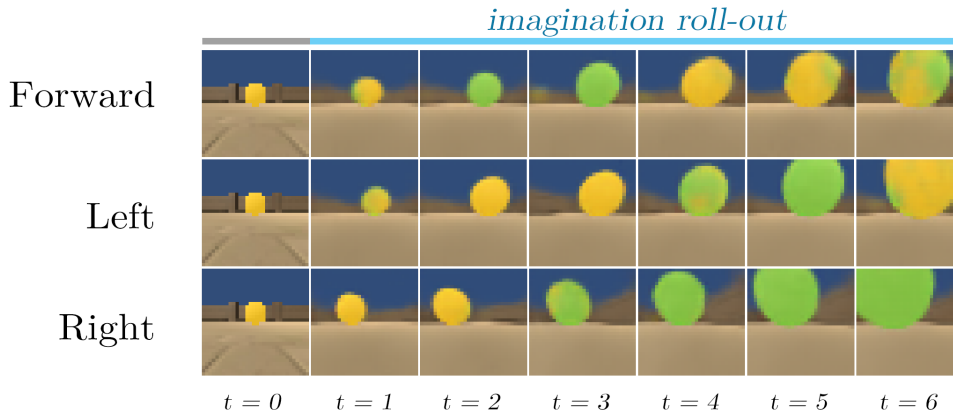


Figure 17: **Action-conditioned imagination roll-outs without action encoding.** The initial TEM transition model struggled to learn action-conditioned predictions. It can be seen that the model does not differentiate between different actions very well.

Therefore, we implement a simple heuristic that is enough to provide the system with the information about the entire trajectory taken by the agent. More concretely, the purpose of the heuristic is to summarise a sequence of actions taken from one memory to reach the next one. A sequence of actions, $A = \{a_{\tau_1}, a_{\tau_1+1}, \dots, a_{\tau_1+(N-1)}\}$, takes the agent from state s_{τ_1} to state s_{τ_2} , where the time between these states $\tau_2 - \tau_1 = N$, and $a \in \{a_{\text{forward}}, a_{\text{right}}, a_{\text{left}}\}$. We employ polar coordinates relative to the agent's position in Cartesian coordinates at time τ_1 and perform iterative updates of its position after every action until the time-step of the next episodic memory.

Agent's position, p , is represented by a vector of x- and y-coordinates,

$$p_t = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}_{t=\tau_1} \quad (32)$$

Polar coordinates are used to update position, p , after every action. This is done by holding information about the agent's orientation, θ , which is changed whenever the agent makes a turning action, and calculating the next position in Cartesian coordinates whenever an agent takes a forward action using,

$$p_{t+1} = p_t + \begin{bmatrix} \sin \theta \\ \cos \theta \end{bmatrix} \quad (33)$$

Finally, when the agent's final position is calculated, we retrieve an angle ϕ , as shown in Figure 18. This angle is used to decide on the action that summarises the trajectory using

$$a = \begin{cases} a_{\text{forward}} & |\phi| \leq 22.5^\circ \\ a_{\text{right}} & 22.5^\circ < \phi < 180^\circ \\ a_{\text{left}} & -22.5^\circ > \phi \geq -180^\circ, \end{cases}$$

excluding special cases. These include situations when the agent does not take forward actions, for which the inference of a summarising action is trivial. A general example of encoding a sequence of actions is shown in Figure 18. Although this heuristic showed improvement, trajectory encoding is one of the most limiting parts of the TEM module and is a promising direction for further research.

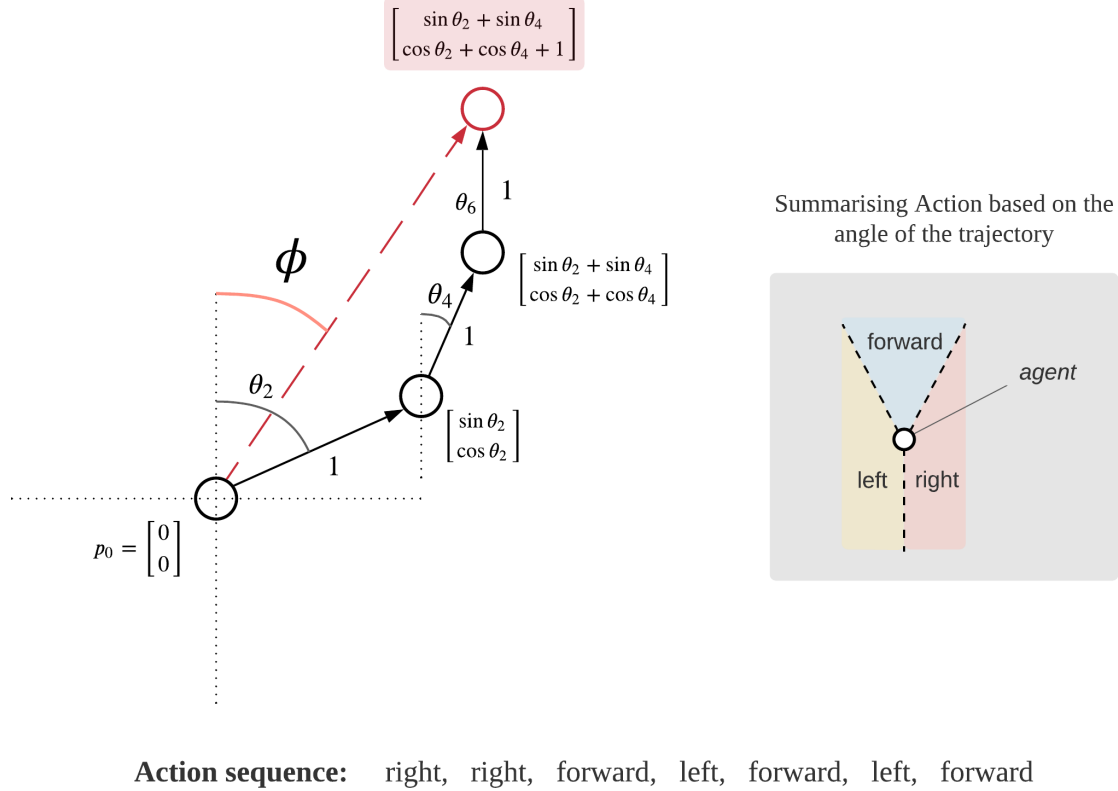
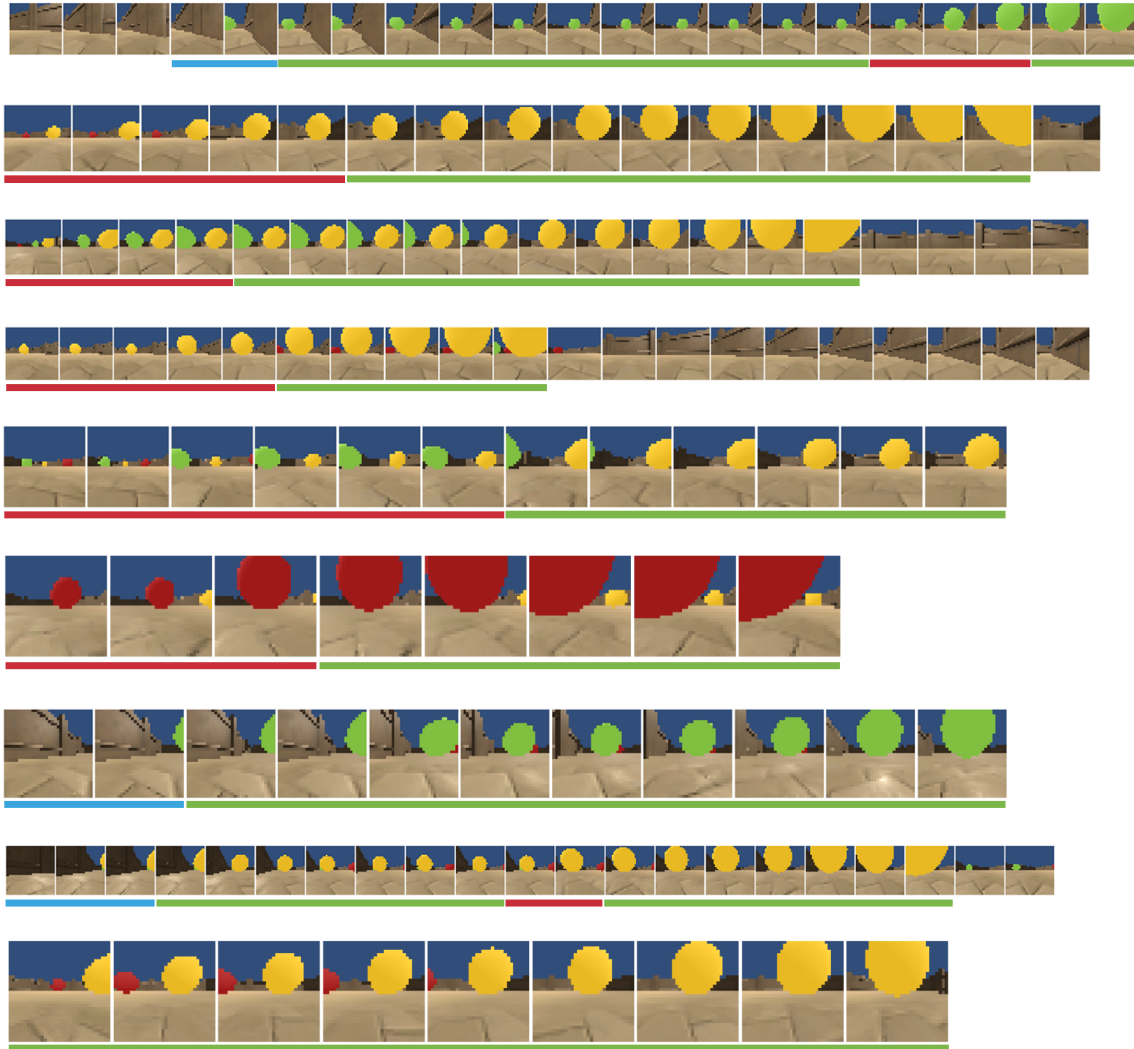


Figure 18: Calculating angle ϕ , in order to summarise a sequence of actions.

5.3 Memory Accumulation Inspection

Figure 19 shows examples of the recorded Sub-sequences during memory accumulation. It is useful to study them in the context of the described characteristics our TEM transition model was intended to have. First, we see quite a distinct slowing down and speeding up of the timescale defined by these sequences (see red and green underlining). In particular, the model tends to skip observations in situations when objects are far away, and increase their frequency as objects get closer. Second, it is quite often that surprising events occur in the Sub-sequences (see blue underlining). These events are important, as they provide the TEM transition model with the ability to learn to predict interesting events given the current state. For instance, in the shown Sub-sequences, the surprising events are all in situations when an agent turns around from a wall and observes a green or yellow sphere. As we discussed in the Motivating Scenario 2, if the TEM transition model learns to predict a green or a yellow sphere in its imagination roll-out, it could provide the agent with an

enhanced ability to take the correct action given its sub-optimal state, in which the baseline agent would be prone to get stuck.



Examples of:

- Fast subjective timescale
- Slow subjective timescale
- Surprising events

Figure 19: **Examples of recorded Sub-sequences.** The Sub-sequences are characterised by the slowing and speeding timescale – depending on the context – and by the presence of surprising events.

5.4 TEM Imagination Roll-out Inspection

In this section, we analyse the qualitative results by studying imagination roll-outs produced by the system under the TEM transition model. Importantly, we inspect the roll-outs with respect to the two characteristics outlined in Section 5.1.

Subjective Timescale. First, we perform simple roll-outs in controlled configurations for each type of sphere. Specifically, we test the model for its ability to speed up and slow down the timescale depending on how far the objects are, as well as whether the action encoding allows the model to learn action-conditioned predictions (the problem which was shown in Figure 17). The retrieved roll-outs are shown in Figure 21.

The displayed TEM imagination roll-outs indicate that the model performs longer temporal jumps when an object is farther away. This can be seen more clearly when compared with the ground-truth roll-out shown in Figure 20. Similarly, the transition model tends to slow down the subjective timescale when the object is closer. This is in line with what we expected the model to do, and is explained by the fact that the system deemed states closer to spheres more interesting during training. This trend appears to be systematic and can be confirmed by studying random roll-outs shown in Figure 23 later in the section.

It can also be observed that the action encoding resulted in significantly improved action-conditioned predictions. It can be seen that forward predictions result in the spheres getting closer, while left- and right-action predictions move the spheres in the correct directions. Interestingly, the left and right predictions for the red sphere diverge from the behaviour of the model for yellow and green spheres. Specifically, the red sphere is much quicker to disappear from imagination roll-outs. This can indicate that the model may have not trained enough on these transitions, given that an active inference agent tends to avoid it.

One other interesting observation is the significant entanglement of the green and yellow spheres. This is likely due to the similarity of their colours, forcing the autoencoder’s bottleneck to keep their latent clusters close together.

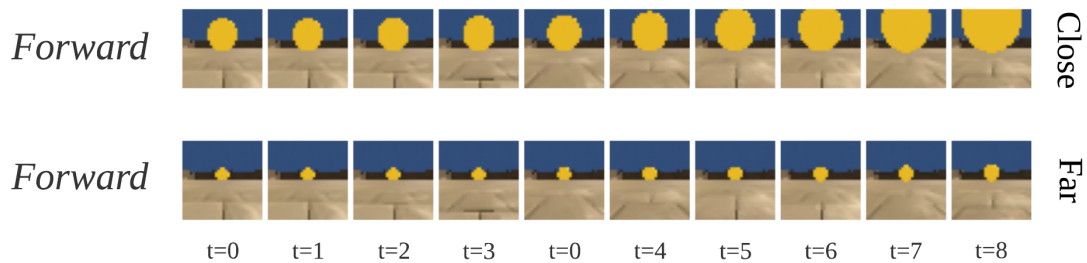


Figure 20: **The ground-truth observations** given a forward action with a yellow sphere. This is useful to study with respect to the subjective timescale learned by the TEM model – Figure 21

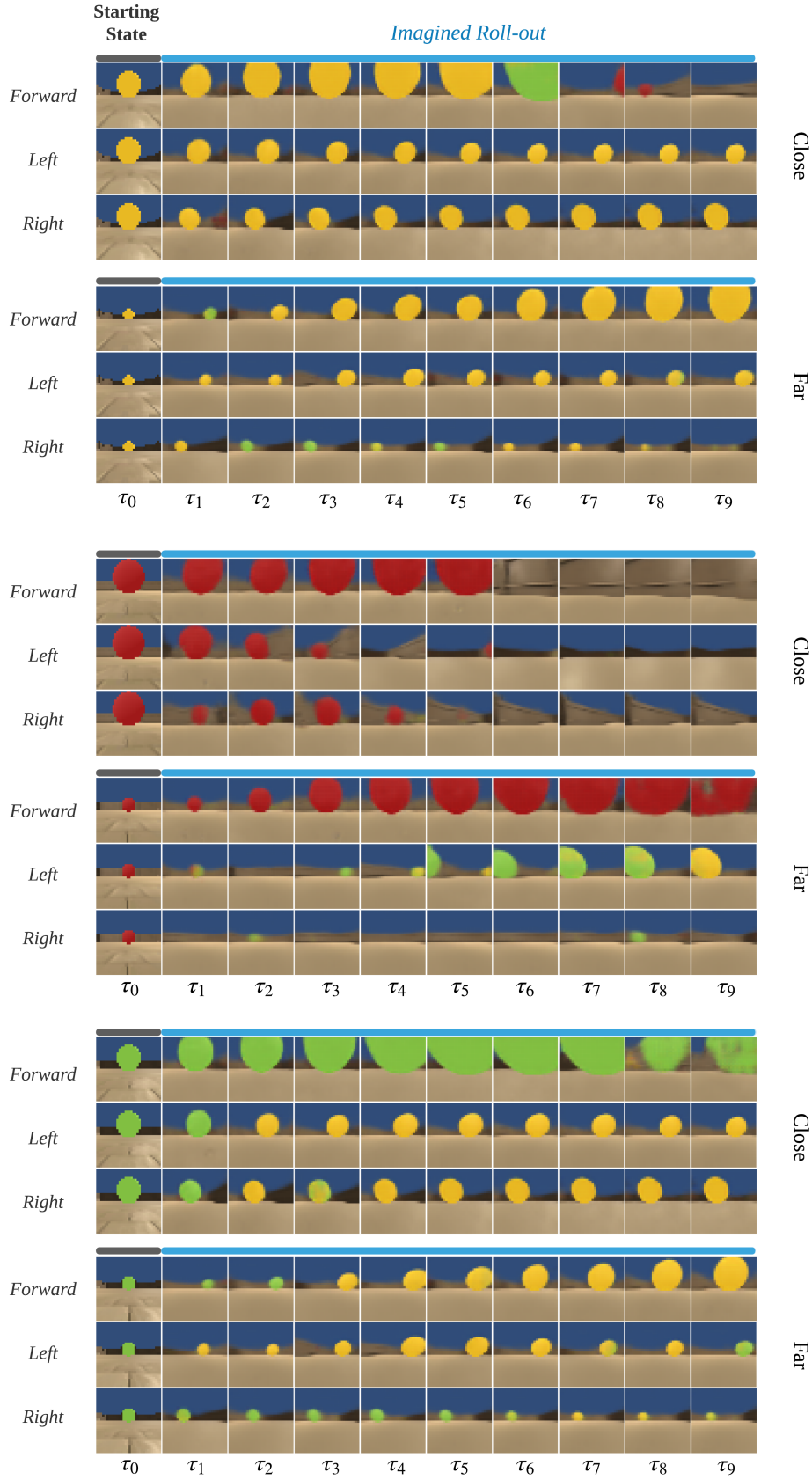


Figure 21: **TEM imagination roll-outs in controlled environments.** The model learns to predict further into the future, when objects are far away, and vice versa. Action-conditioned predictions are also significantly improved with action encodings introduced in Section 5.2

Imagining objects. Next, we investigate the model’s ability to imagine objects in the roll-outs. As discussed, this property would be conducive for the agent’s ability to make more exploratory decisions, when it is stuck in a sub-optimal state. It would also be a natural consequence of the training procedure we employed, in which the system is trained on entire sequences of *surprising*¹¹ and *temporally-dependent* states. Figure 22 shows examples generated in the first 50 random imagination roll-outs. We stress that these samples were not selectively picked from a large number of roll-outs, rather they frequently occurred in the produced sequences.

From these roll-outs, it becomes evident that the model is able to imagine objects given an initial, uninteresting state. Moreover, the observations produced by the model are entirely plausible given the path the agent is taking and the context it finds itself in. This indicates that the TEM transition model does indeed produce semantically meaningful predictions. Furthermore, it is pertinent to note that the roll-outs comply with the physics of the environment. One aspect that could be indicative of this is if the shape and brightness of the walls change in accordance with the true physics of the environment; indeed, this is the case in the provided roll-outs. This point is crucial, as it potentially refutes the hypothesis that these imagined objects were predicted at random. If the model is able to predict a plausible evolution of its surroundings given its action, then it is, loosely speaking, aware of its environmental context. And given the intentional design of our model, the appearance of these objects indicates that the model predicts these states as *likely, given the initial observation*. It is also similar to how a human agent would treat the initial, uninteresting observation – if there are no objects in the current frame, then they must be elsewhere. This is in stark contrast to the original, slow transition model that is not able to imagine objects that are not present in the initial frame. See Figure 37 in Appendix D, which shows imagination roll-outs from the same initial states but using the objective-timescale transition model.

Additional Examples. Lastly, before moving to the quantitative results, we provide a few extra randomly generated roll-outs in Figure 23. Again, we note that these were picked from the same batch of 50 randomly-generated roll-outs. Here, we similarly show that the model is able to predict a much faster approach to spheres and slow its prediction timescale when they are close. Similarly, the model is able to correctly predict the expected future observations given an action in the respective direction. These are exemplified by the second and last sets of roll-outs in the figure. One additional interesting observation that can be made from studying these samples is the presence of disappearing and later re-appearing objects. We believe, this can be attributed to the architectural nature of the TEM model. Specifically, the TEM transition model is an autoregressive model, which means it encodes information about the preceding states. If a sphere appears far away, the decoder is unlikely to reconstruct it in the image given the insignificant signal in the latent space (i.e. the bottleneck of the autoencoder). However, this small signal is likely to persist through time and later increase as the TEM model predicts the sphere to get closer and bigger. As a result, the TEM model has better representational power with respect to *object permanence*,

¹¹Again, we note that *surprising* merely implies events with transition-model free energies above a certain threshold.

which is one of the major research focuses in the Animal-AI environment.

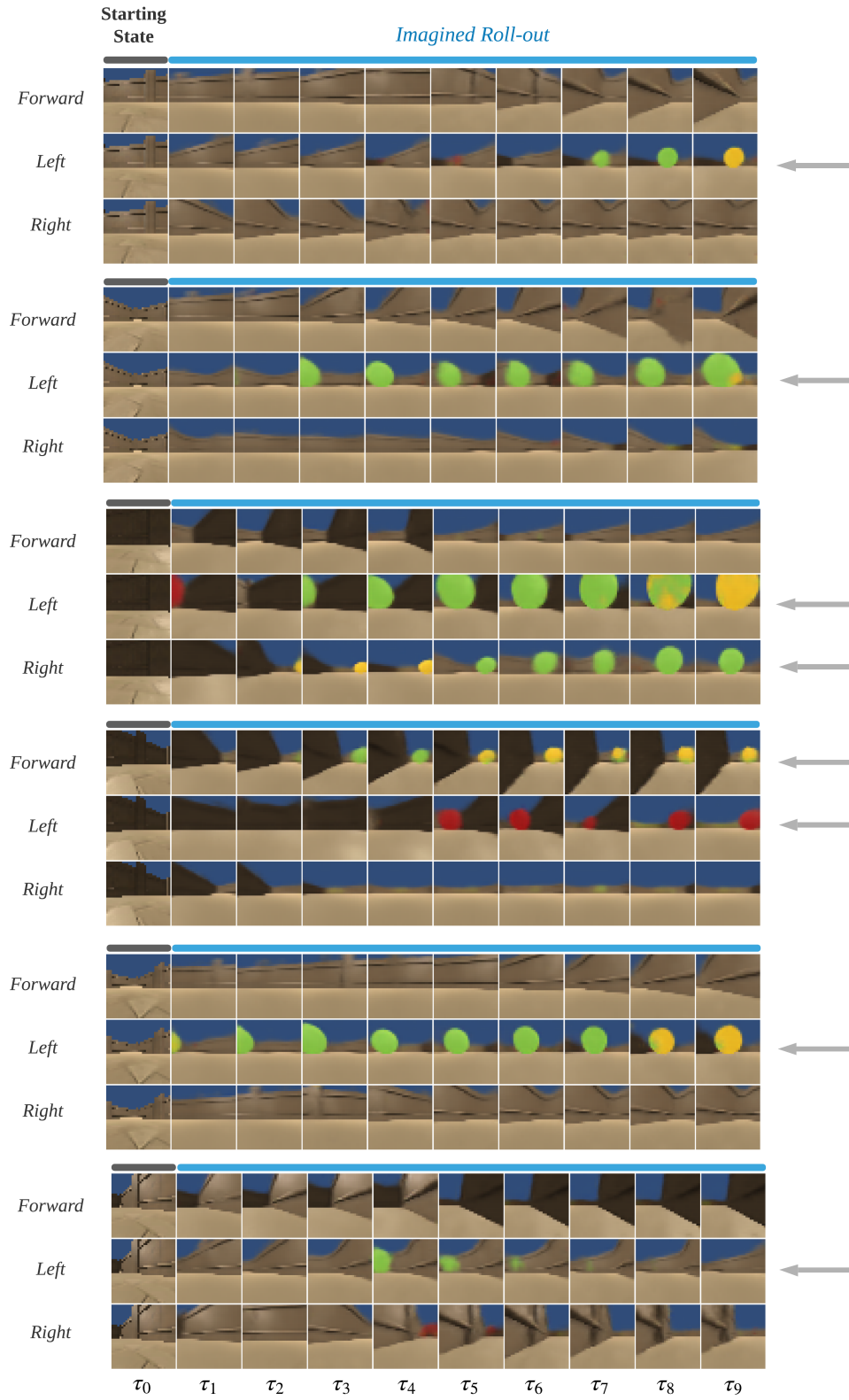


Figure 22: **TEM model can imagine objects from uninteresting states.** Arrows indicate roll-outs with imagined objects. Given the initial state and the imagined observations, the system will be less prone to go to a wall.

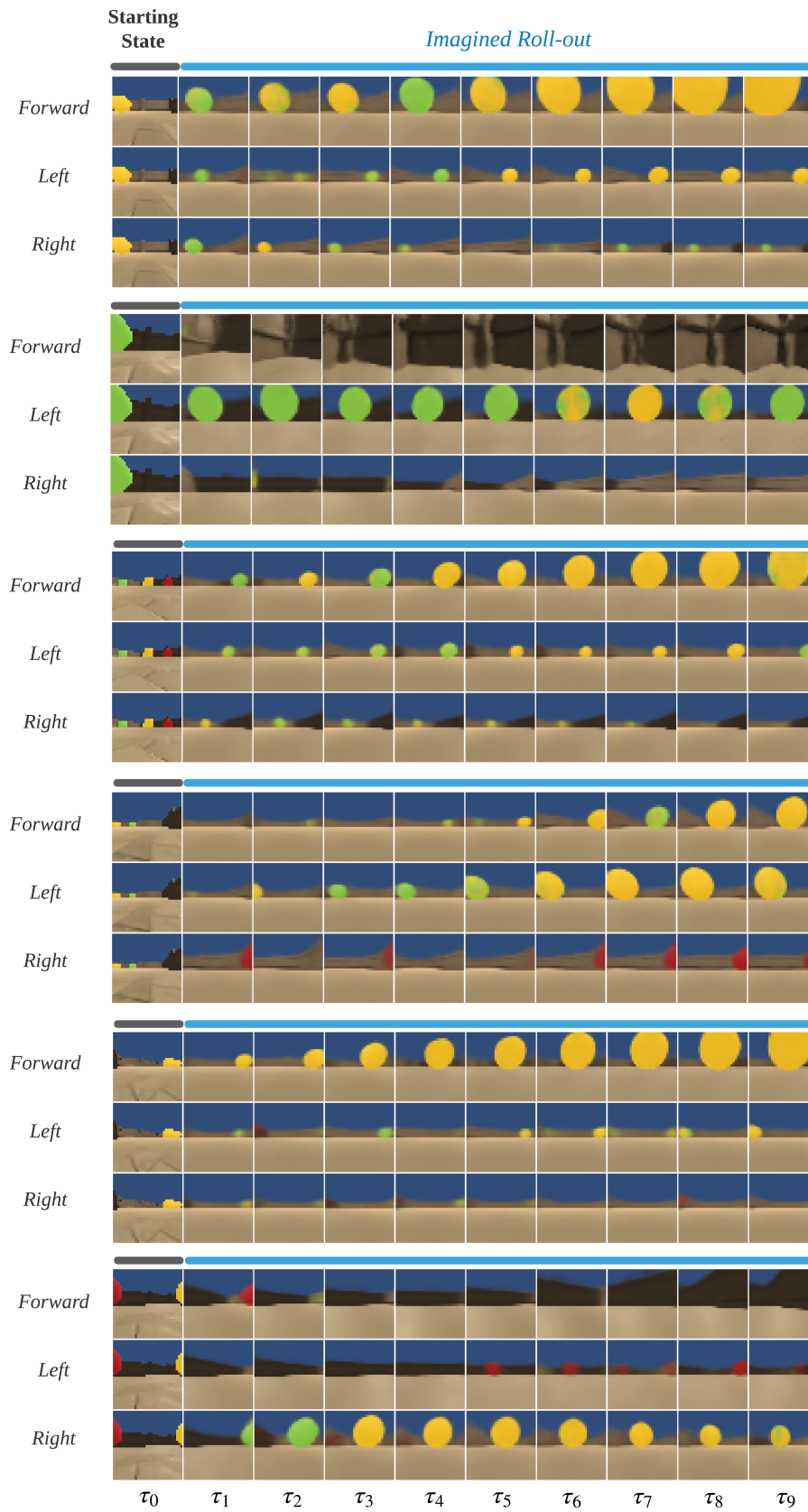


Figure 23: Random roll-outs generated with the TEM transition model.

5.5 Experimental Results

In this section, we investigate the quantitative performance of an agent equipped with the TEM transition model. We first compare the TEM agent’s performance with the PRB agent from Section 4, which we take as the baseline, using the same experimental setup. Second, we test the system on a range of tasks analogous to those from the Animal-AI 2019 competition [12].

As mentioned, to compare the TEM agent to our baseline agent with PRB, we employ the same experimental setup used to test on-line vs. PRB agents in Section 4.3. Recall that we did not implement MCTS then due to the agents’ tendency to spend significant amount of time at the walls. Instead, we employed model-predictive control (MPC) that would allow the agent to re-evaluate its plan after every taken action. However, MPC struggles from being computationally-expensive, and to which MCTS offers an efficient alternative. Because the use of MCTS affords computational savings and biological plausibility, its implementation in the final system is highly desirable. We show that our system equipped with TEM transition model and the efficient MCTS algorithm outperforms the PRB agent with MPC.

Importantly, we do not use the habitual network in MCTS to perform early action selection¹²; thus, the model must rely entirely on the calculated values of the expected free energy with respect to its generative model. As a result, the habitual network is used *only* as a prior to search the tree of possible trajectories. This allows us to better understand the performance of an agent’s generative model, of which the TEM module is a major part. Appendices C.1 and C.2 include the MCTS hyperparameters chosen for the experiments. In what follows, we compare the TEM agent equipped with MCTS (TEM-MCTS) against *both* the PRB baseline agent using MCTS (PRB-MCTS) and MPC (PRB-MPC) for planning.

We stress that the compared TEM and PRB agents *share the same autoencoder* – both the architecture and the network weights. Thus, the performance improvements can more confidently be attributed to the main distinguishing component – the transition model.

Figures 24 and 25 summarise the experimental results. It can be observed that the TEM-MCTS agent outperforms the other systems in acquiring more rewards within the 100,000 steps. In particular, we note that the TEM-MCTS agent showed significant improvement against the baseline PRB-MCTS. This substantiates the analysed qualitative characteristics of the TEM module in Section 5.4 and provides empirical support for the hypothesised improvements of the system. Similarly, we also show that TEM-MCTS model retrieves more cumulative reward than the PRB-MPC agent, which uses a computationally expensive planning procedure, as mentioned earlier. Specifically, our agent achieves a higher cumulative reward in less than half the time, ~ 6 hours, compared to ~ 14 hours for PRB-MPC. However, TEM-MCTS receives more negative rewards, which is a consequence of the fact that re-planning of the agent’s actions occurs only after the entire pre-planned trajectory has been executed. Therefore, MPC-based system is more able to avoid red spheres, as the re-evaluation of its trajectory happens after every step.

¹²Recall that, in the original system with MCTS, an agent employed a habitual network to select an action, if the habitual network was confident about its decision above a certain threshold, T_{dec} .

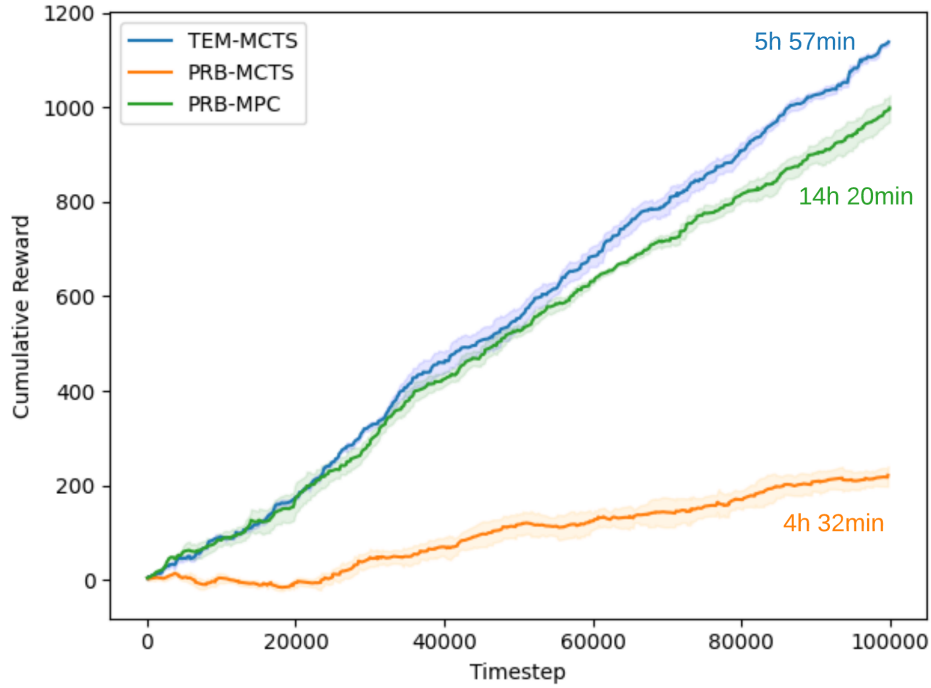


Figure 24: **Performance of the TEM agent compared to the baseline agents.** The figure shows the mean cumulative rewards collected by the agents (with one standard deviation over 5 runs), including the PRB agent with model-predictive control from Section 4.3. It can be seen that the TEM-MCTS agent shows improved performance even when compared with PRB-MPC.

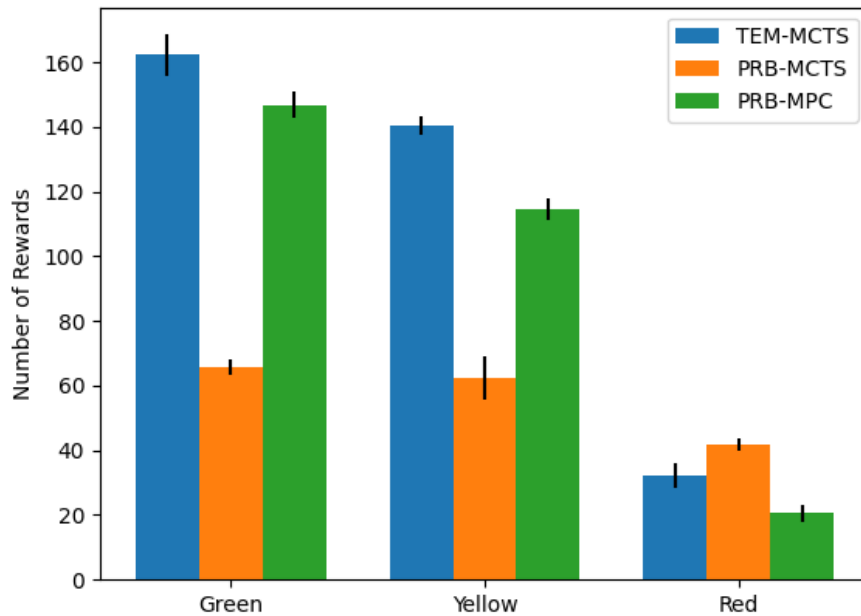


Figure 25: **Mean number of rewards by category.** TEM-MCTS agent collects more green and yellow rewards than PRB-MCTS and PRB-MPC agents. However, PRB-MPC collects fewer red rewards than TEM-MCTS, which is likely related to the ability of the former to evaluate its action after every step. Black lines indicate one standard deviation over 5 runs.

Specific Configurations Testing. Finally, we compare the performance of TEM-MCTS against PRB-MCTS on a number of configuration maps analogous to those from the 2019 Animal-AI competition [12]. These tasks are characterised by the very limited episode time (250 steps compared to 500 in our previous tests) and allow us to better understand the agent’s performance given a number of fixed configurations. Both agents are tested 10 times on each configuration. Specifically, we use basic tasks like:

1. Green sphere ahead, Figure 26
2. Positional variations, Figure 27
3. Exploration tasks, Figure 28
4. Multiple foods tasks, Figure 29

As during training, the sphere size was set to 5, where possible.

The results in the figures below show that the agent with the TEM module significantly improves against its baseline that uses the objective-timescale transition model.

Green Ahead. TEM system is able to successfully retrieve green rewards in *Green Ahead* tasks (Figure 26). In contrast, the baseline agent begins to struggle with the tasks, once the initial distance between the sphere and the agent increases. Once again, this confirms the observed temporal jumps that the TEM agent can perform when a sphere is farther away, thus incentivising the agent to go forward and not turn away from it.

Variations. The TEM agent similarly better solves *Variations* tasks (Figure 27). These tasks require an agent to choose the correct direction of movement from the beginning, as well as approach a relatively distant sphere in a limited amount of time.

Exploration. These tasks were significantly more challenging for both of the agents (Figure 28). However, the TEM agent still performs better than the baseline agent. By inspecting the actual behaviour of the TEM agent, we found that it always turned around and was almost never stuck at walls. This particular type of test provided more significant grounds to believe that the TEM agent’s ability to imagine objects is able to remove it from the sub-optimal state at a wall. In contrast, we emphasise a severely limited performance of the baseline agent, which struggled with this very problem. Still, the TEM agent was often not fast enough to reach the reward. Furthermore, we noticed that the TEM system was more prone to turn right for exploration, thus performing worse on tasks where the correct choice was to go left. Additionally, we note that both systems generally performed better on yellow spheres, as their contrasting colour was better represented in the latent space of the autoencoder and could be more easily recognised in the distance.

Multiple Foods. Multiple food tasks required an agent to be much more exploitative and quick, given a fairly densely populated map (Figure 29). We observed the TEM agent to be much better at going straight to a yellow reward once observed. This can be attributed to its ability to imagine a much faster approach, which made its choices greedier and more obvious during MCTS.

On the other hand, over the course of this testing, it became clear that the overall performance of the entire system is still quite limited. In particular, we emphasise two hindering factors – the observation resolution and the autoencoder. For training, we used observations of 32x32 pixels, which significantly restricts what the agent is able to see. For instance, smaller spheres can only be seen when an agent approaches them really closely, where as a relatively large sphere will take only *one* pixel of the entire observation, if far away. Furthermore, the agent is still largely short-sighted, as the autoencoder is poor at encoding and reconstructing small objects present in an image. These and other limitations of the system will be discussed in more detail in Section 6.




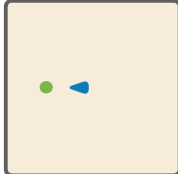


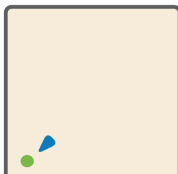


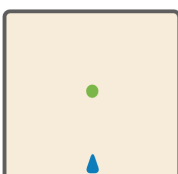


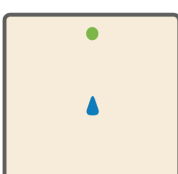


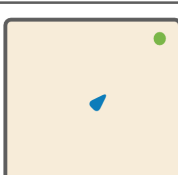


Task Type	Sphere Size	Distance to Sphere	Configuration Map	EM-MCTS	PRB-MCTS
<i>Green Ahead I</i>	1	2			
<i>Green Ahead I</i>	2	3			
<i>Green Ahead I</i>	2	1.5			
<i>Green Ahead II</i>	5	15			
<i>Green Ahead II</i>	5	15			
<i>Green Ahead II</i>	5	15			

Figure 26: *Green Ahead*: reward ahead.

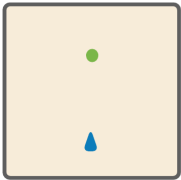


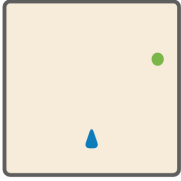


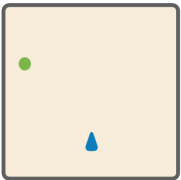
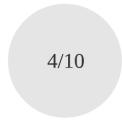

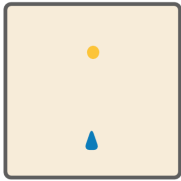

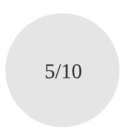
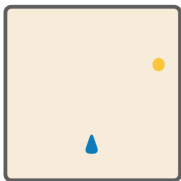

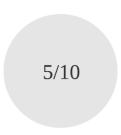
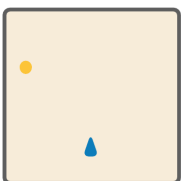

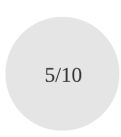
<i>Task Type</i>	<i>Sphere Size</i>	<i>Distance to Sphere</i>	<i>Configuration Map</i>	<i>EM-MCTS</i>	<i>PRB-MCTS</i>
Variations I	5	16			
Variations I	5	30			
Variations I	5	30			
Variations II	5	16			
Variations II	5	30			
Variations II	5	30			

Figure 27: *Positional Variations*: reward ahead but in different positions.

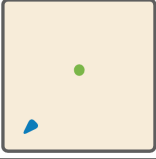

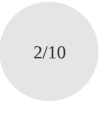
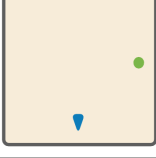


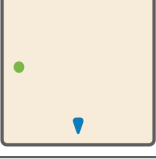


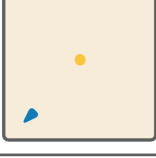

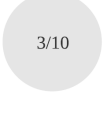
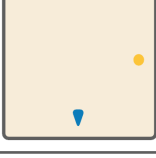


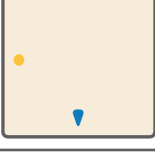

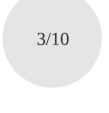
Task Type	Sphere Size	Distance to Sphere	Configuration Map	EM-MCTS	PRB-MCTS
Exploration I	5	18			
Exploration I	5	24			
Exploration I	5	24			
Exploration II	5	18			
Exploration II	5	24			
Exploration II	5	24			

Figure 28: *Exploration*: turn around and get a reward.



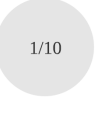


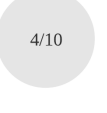

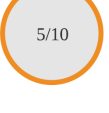
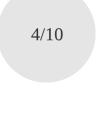
Task Type	Sphere Size	Pass Condition	Configuration Map	EM-MCTS	PRB-MCTS
Multiple Yellow I	5	Get two			
Multiple Yellow I	5	Get two			
Multiple Yellow I	5	Get two			

Figure 29: *Multiple Foods*: get at least two yellow foods.

6 Evaluation and Future Work

In this section, we provide a critical evaluation of the entire system, as well as its newly-introduced constituent parts. We start by briefly summarising the main contributions and results of this project. Next, we integrate the discussion about limitations with future research directions, as the two are inseparably linked together.

6.1 Contributions

The main theme of the project was the integration of episodic memory into an active inference agent. We proposed two methods based on the weak and strong forms of episodic memory to improve the performance of the state-of-the-art baseline agent from Fountas et al. [10].

Prioritised Replay Buffer. We reconciled the framework of active inference with the pre-dominantly RL technique of prioritised sampling, which, to our knowledge, has never been attempted before. Specifically, we performed a thorough analysis of the different prioritisation strategies based on the values of the free energies, and empirically validated the improvement in the agent’s generative model, and as a result of the entire system.

TEM Module. We proposed a novel approach to learning a transition dynamics model with the use of episodic memory, which showed significant improvement in the active inference agent’s performance on a number of tasks. Inspired by the problems of inaccurate and inefficient long-term predictions in model-based RL and the recent neuroscience literature on episodic memory and human time perception, we merged ideas from the different fields into one new technique of learning a forward model. We demonstrated empirical evidence of the system’s improved performance over its baseline counterpart, as well as exemplified some important characteristics of the newly-devised system, such as the abilities to imagine objects or to vary the speed of the prediction timescale. Furthermore, the application of our technique is not limited to active inference, and has the potential for integration in model-based reinforcement learning.

6.2 Limitations and Future Work

Overall System. The most striking limitation of the entire system is the size of the input observations used to train our agent (only 32x32 pixels), which is certainly not enough to get a good performance in the environment. This was largely exemplified by the results of the agent on the original Animal-AI competition tests, in which small spheres did not appear in the ground-truth observations even at very close distances. Similarly, even a relatively large sphere would take up only *one* pixel of the entire frame, if placed sufficiently far from the agent. Thus, the model would undoubtedly benefit from increasing the size of its input.

This problem is further exacerbated by the relatively poor performance of the variational autoencoder. Arguably, it is the most critical part of the entire system, acting as a bridge

between the agent’s internal model and the outside environment. In particular, we refer to the agent’s ability to *encode* observations into useful hidden state representations, and its ability to *decode* hidden states into accurate imagined frames. The use of prioritisation with PRB was an attempt to improve the quality of the generative model with respect to the more important states in the environment, as well as to learn more object-centric representations. The results showed visible improvements; however, the system is still limited by the intrinsic architecture of the autoencoder. For instance, the model struggled to represent multiple or small objects in the same frame (Figure 30) and the proposed latent space regularisation is still not directly conducive to learning object-centric representations. For this reason, we believe our system would benefit greatly from the more advanced autoencoding architectures, such as variational autoencoders with discrete latent space [113] or slot-based mechanisms [114].

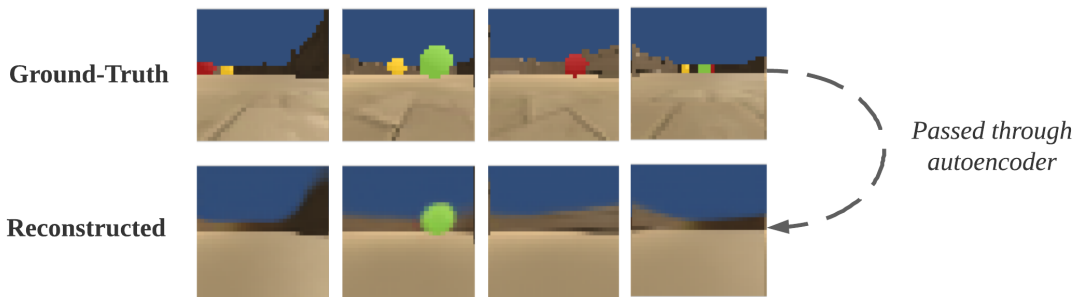


Figure 30: Autoencoder struggles to reconstruct frames with small or multiple objects.

Furthermore, we stress the relative simplicity of the test configurations. Although, the setup was specifically designed to be simple and sparse in rewards, we suggest further experiments using more complicated tasks.

Prioritised Replay Buffer. Although the use of PRB yielded satisfactory improvements – both qualitative and quantitative – a more rigorous analysis of its effects is required. Specifically, in Section 4.2, we motivated the choice of the priority metric by inspecting the sorted buffer of transitions and providing justifications for the most useful priority metric. However, the explanations are limited to qualitative evidence and would benefit from more mathematical and theoretically-grounded reasons.

TEM Agent. TEM transition model has a number of limitations that would, if addressed, yield a better performance. Firstly, the action-encoding heuristic significantly constraints the model’s applicability to more complex environments. This becomes evident if we consider environments where the order in which the actions are taken matters (e.g. going behind a wall). The current action encoding does not take this into account, resorting to a very simple summarisation technique instead. Thus, a promising direction of research is to come up with a more principled way to encode entire trajectories, which would additionally include information about action ordering. Here, segment-based approaches [14] discussed in Section 2.2.3 are of particular interest.

Secondly, the threshold for forming episodic memory was chosen manually by inspecting the distribution of surprise values in the buffer. Because the threshold is fixed, memory accumulation can have lengthy periods of no recorded memories. In turn, this limits the system’s ability to learn the physics of the environment and can confuse it given that no clear temporal relation between two events can be traced. Here, the system could benefit from a dynamic, exponentially-decaying threshold as implemented in Fountas et al. [105], which allows for a more temporally-equidistant formation of memories. For instance, in the absence of anything surprising, the threshold would slowly decay, allowing the system to form a memory. However, more work would need to be done to preserve the slowing and speeding of the subjective timescale, which is a necessary characteristic of the devised transition model.

Thirdly, the TEM model’s ability to imagine objects could potentially be related to its autoregressive nature, as opposed to the characteristics of the collected memories. Therefore, we suggest further tests against an active inference agent equipped with an objective-timescale autoregressive dynamics model.

Next, we believe the system would greatly benefit from imagination roll-outs given the entire history of states observed in an episode. Although the current model is autoregressive, its initial hidden state in a roll-out is taken as the *current state* of the agent in an environment. Specifically, the zeroth (h_0) and the initial (h_1) LSTM hidden states in a roll-out are actually:

$$h_0 = \mathbf{0}, \quad (34)$$

$$h_1 = f_{\theta_h}(\tilde{s}_1, a_1, h_0), \quad (35)$$

where \tilde{s}_1 is the *last state observed* by the agent in the environment, and a_1 is the first action taken in the imagination roll-out. Here, the tilde above the state variable is used to denote states that were *observed* in the environment, rather than imagined.

All the next LSTM hidden states are calculated via iterative application of function f_{θ_h} discussed in Section 5.2:

$$h_t = f_{\theta_h}(s_t, a_t, f_{\theta_h}(s_{t-1}, a_{t-1}, h_{t-2})), \quad (36)$$

Importantly, we see that the roll-out is performed only with respect to the *last*-observed state and all the following imagined states. Ideally, however, we want the agent to make imagination roll-outs with respect to *all* the preceding states it observed in the environment, such that:

$$h_0 = h_{episode}, \quad (37)$$

$$h_1 = f_{\theta_h}(\tilde{s}_1, a_1, h_0), \quad (38)$$

where $h_{episode}$ is the LSTM hidden state calculated from the sequence of all previously observed states in an episode. This way, we expect the transition model to have a better predictive power, given that it would have more information about the agent’s preceding observations.

Lastly, the current implementation of the TEM agent relies on having a separate objective-timescale transition model. As we saw, this transition model is used to collect episodic memories that are later used to train the TEM transition model. Having two separate networks working asynchronously may not be the most computationally-efficient and elegant way to achieve the goals we set out. An appealing future direction of research could include having a single autoregressive transition model that slowly transitions from training on an objective timescale to training on a subjective timescale, as the memory formation proceeds.

7 Conclusion

In this project, we introduced two ways of integrating episodic memory into an active inference agent. In particular, we implemented two systems that effectively addressed the questions outlined in the beginning of Sections 4 and 5:

Section 4: Given the baseline architecture outlined in Section 3, what data should our on-line agent train on?

Section 5: Can we come up with a principled way to learn the transition dynamics model over a more useful, subjective timescale of the environment?

Section 5: Can we learn a transition dynamics model that is capable of imagining ‘novel’ observations in the future?

We started by using a weak form of episodic memory, in order to have better control of what our generative model learns. We integrated the idea of prioritised experience replay with the framework of active inference, which proved to be an effective way to improve the agent’s performance. Specifically, we found that prioritising data with higher values of transition-model free energy was conducive for learning the more important features of the environment.

In the second part of the report, we considered how the temporal information of the memories could be employed to answer the latter two questions. As a result, we developed a novel technique for learning a transition dynamics model over a subjective timescale, which exhibited interesting characteristics useful for solving tasks in the Animal-AI environment. We found that the learned subjective timescale provided the agent with better means of making more informed decisions in the environment. We similarly showed that the new model was capable of imagining novel future scenarios.

Our work has once again demonstrated the practical value and the extent of applications of episodic memories. Furthermore, by applying techniques that have been primarily used in the field of RL to active inference, we are slowly making steps to bridge the gap between the two fields, which in reality share lots of similarities. Additionally, we believe this project can provide strong grounds for future research on learning subjective-timescale transition models. As mentioned, these models more closely mimic the characteristics of human time-perception and can provide more flexible, informative and computationally-efficient predictions of the future.

References

- [1] Karl J Friston, James Kilner, and Lee M. Harrison. A free energy principle for the brain. *Journal of Physiology-Paris*, 100:70–87, 2006.
- [2] Karl J. Friston. A free energy principle for a particular physics. *arXiv: Neurons and Cognition*, 2019.
- [3] Thomas Parr and Karl J Friston. Generalised free energy and active inference. *Biological Cybernetics*, 113:495 – 513, 2019.
- [4] Noor Sajid, Philip J. Ball, and Karl J. Friston. Active inference: demystified and compared. *arXiv: Artificial Intelligence*, 2019.
- [5] Tingwu Wang, Xuchan Bao, Ignasi Clavera, Jerrick Hoang, Yeming Wen, Eric Langlois, S. Zhang, Guodong Zhang, Pieter Abbeel, and Jimmy Ba. Benchmarking model-based reinforcement learning. *ArXiv*, abs/1907.02057, 2019.
- [6] David R Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *NeurIPS*, 2018.
- [7] Johannes B Mahr and G. Csibra. Why do we remember? the communicative function of episodic memory. *The Behavioral and brain sciences*, pages 1–93, 2017.
- [8] Alice Mason, S. Farrell, P. Howard-Jones, and Casimir J H Ludwig. The role of reward and reward uncertainty in episodic memory. *Journal of Memory and Language*, 96:62–77, 2017.
- [9] D. L. Greenberg and M. Verfaellie. Interdependence of episodic and semantic memory: evidence from neuropsychology. *Journal of the International Neuropsychological Society : JINS*, 16 5:748–53, 2010.
- [10] Zafeirios Fountas, Noor Sajid, Pedro A. M. Mediano, and Karl J. Friston. Deep active inference agents using monte-carlo methods. *ArXiv*, abs/2006.04176, 2020.
- [11] Benjamin Beyret, José Hernández-Orallo, Lucy G Cheke, Marta Halina, Murray Shanahan, and Matthew Crosby. The animal-ai environment: Training and testing animal-like artificial cognition. *ArXiv*, abs/1909.07483, 2019.
- [12] Matthew Crosby, Benjamin Beyret, Murray Shanahan, José Hernández-Orallo, Lucy Cheke, and Marta Halina. The animal-ai testbed and competition. volume 123 of *Proceedings of Machine Learning Research*, pages 164–176, Vancouver, CA, 08–14 Dec 2020. PMLR.
- [13] Franccois Chollet. On the measure of intelligence. *ArXiv*, abs/1911.01547, 2019.
- [14] Nikhil Mishra, Pieter Abbeel, and Igor Mordatch. Prediction and control with temporal segment models. *ArXiv*, abs/1703.04070, 2017.

- [15] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *ICML*, 2000.
- [16] Sébastien Racanière, Theophane Weber, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adrià Puigdomènech Badia, Oriol Vinyals, Nicolas Manfred Otto Heess, Yujia Li, Razvan Pascanu, Peter W. Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. Imagination-augmented agents for deep reinforcement learning. In *NIPS*, 2017.
- [17] Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566, 2018.
- [18] Máté Lengyel and Peter Dayan. Hippocampal contributions to control: The third way. In *NIPS*, 2007.
- [19] Daniel Kahneman. Thinking, fast and slow. 2011.
- [20] Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *NIPS*, 2014.
- [21] Manuel Watter, Jost Tobias Springenberg, Joshka Boedecker, and Martin A. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *NIPS*, 2015.
- [22] J. Andrew Bagnell and Jeff G. Schneider. Autonomous helicopter control using reinforcement learning policy search methods. *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164)*, 2:1615–1620 vol.2, 2001.
- [23] Pieter Abbeel, Morgan Quigley, and Andrew Y. Ng. Using inaccurate models in reinforcement learning. In *ICML '06*, 2006.
- [24] Sergey Levine and Pieter Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *NIPS*, 2014.
- [25] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *J. Mach. Learn. Res.*, 17:39:1–39:40, 2016.
- [26] Vikash Kumar, Emanuel Todorov, and Sergey Levine. Optimal control with learned local models: Application to dexterous manipulation. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 378–383, 2016.
- [27] Emanuel Todorov and Weiwei Li. A generalized iterative lqg method for locally-optimal feedback control of constrained nonlinear stochastic systems. *Proceedings of the 2005, American Control Conference, 2005.*, pages 300–306 vol. 1, 2005.

- [28] Marc Peter Deisenroth and Carl E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *ICML*, 2011.
- [29] Jonathan Ko, Daniel J. Klein, Dieter Fox, and Dirk Hähnel. Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 742–747, 2007.
- [30] D. Silver, Julian Schrittwieser, K. Simonyan, Ioannis Antonoglou, Aja Huang, A. Guez, T. Hubert, L. Baker, Matthew Lai, A. Bolton, Yutian Chen, T. Lillicrap, F. Hui, L. Sifre, George van den Driessche, T. Graepel, and Demis Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.
- [31] Ian Lenz, Ross A. Knepper, and Ashutosh Saxena. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*, 2015.
- [32] Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I. Jordan, Joseph E. Gonzalez, and Sergey Levine. Model-based value expansion for efficient model-free reinforcement learning. 2018.
- [33] G. Brockman, Vicki Cheung, Ludwig Pettersson, J. Schneider, John Schulman, Jie Tang, and W. Zaremba. Openai gym. *ArXiv*, abs/1606.01540, 2016.
- [34] Jacob Buckman, Danijar Hafner, George Tucker, Eugene Brevdo, and Honglak Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *NeurIPS*, 2018.
- [35] Thanard Kurutach, Ignasi Clavera, Yan Duan, Aviv Tamar, and Pieter Abbeel. Model-ensemble trust-region policy optimization. *ArXiv*, abs/1802.10592, 2018.
- [36] Gabriel Kalweit and Joschka Boedecker. Uncertainty-driven imagination for continuous deep reinforcement learning. In *CoRL*, 2017.
- [37] Ignasi Clavera, Yao Fu, and Pieter Abbeel. Model-augmented actor-critic: Backpropagating through paths. *ArXiv*, abs/2005.08068, 2020.
- [38] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Manfred Otto Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *CoRR*, abs/1509.02971, 2015.
- [39] Rowan McAllister and Carl Edward Rasmussen. Improving pilco with bayesian neural network dynamics models. 2016.
- [40] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.
- [41] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *NIPS*, 2016.

- [42] Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Learning and policy search in stochastic dynamical systems with bayesian neural networks. *ArXiv*, abs/1605.07127, 2017.
- [43] Ignasi Clavera, Jonas Rothfuss, John Schulman, Yasuhiro Fujita, Tamim Asfour, and Pieter Abbeel. Model-based reinforcement learning via meta-policy optimization. *ArXiv*, abs/1809.05214, 2018.
- [44] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.
- [45] Nan Rosemary Ke, Amanpreet Singh, Ahmed Touati, Anirudh Goyal, Yoshua Bengio, Devi Parikh, and Dhruv Batra. Learning dynamics model in reinforcement learning by incorporating the long term future. *ArXiv*, abs/1903.01599, 2019.
- [46] Alexander Li, Carlos Florensa, Ignasi Clavera, and Pieter Abbeel. Sub-policy adaptation for hierarchical reinforcement learning. *ArXiv*, abs/1906.05862, 2020.
- [47] Huazhe Xu, Yuezhi Li, Yuandong Tian, Trevor Darrell, and Tengyu Ma. Algorithmic framework for model-based reinforcement learning with theoretical guarantees. *ArXiv*, abs/1807.03858, 2019.
- [48] Sergey Levine and Vladlen Koltun. Guided policy search. In *ICML*, 2013.
- [49] Nicolas Manfred Otto Heess, Gregory Wayne, David Silver, Timothy P. Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. *ArXiv*, abs/1510.09142, 2015.
- [50] Arthur G. Richards. Robust constrained model predictive control. 2005.
- [51] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NeurIPS*, 2018.
- [52] Ignasi Clavera, Anusha Nagabandi, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt: Meta-learning for model-based control. *ArXiv*, abs/1803.11347, 2018.
- [53] Justin Fu, Sergey Levine, and Pieter Abbeel. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4019–4026, 2016.
- [54] Shixiang Gu, Timothy P. Lillicrap, Ilya Sutskever, and Sergey Levine. Continuous deep q-learning with model-based acceleration. In *ICML*, 2016.
- [55] Alexander Tschantz, Manuel Baltieri, Anil K. Seth, and Christopher L. Buckley. Scaling active inference. *ArXiv*, abs/1911.10601, 2019.

- [56] Karl J. Friston, Francesco Rigoli, Dimitri Ognibene, Christoph Mathys, Thomas H. B. FitzGerald, and Giovanni Pezzulo. Active inference and epistemic value. *Cognitive Neuroscience*, 6:187 – 214, 2015.
- [57] Karl J. Friston, Thomas H. B. FitzGerald, Francesco Rigoli, Philipp Schwartenbeck, and Giovanni Pezzulo. Active inference: A process theory. *Neural Computation*, 29:1–49, 2017.
- [58] Karl J. Friston, Marco Lin, Christopher D. Frith, Giovanni Pezzulo, J. Allan Hobson, and Sasha Ondobaka. Active inference, curiosity and insight. *Neural Computation*, 29:2633–2683, 2017.
- [59] Karl J. Friston, Richard E Rosch, Thomas Parr, Cathy J. Price, and Howard Bowman. Deep temporal models and active inference. *Neuroscience and Biobehavioral Reviews*, 90:486 – 501, 2017.
- [60] Kai Ueltzhöffer. Deep active inference. *Biological Cybernetics*, 112:547–573, 2018.
- [61] A. W. Moore. Efficient memory-based learning for robot control. 1990.
- [62] Ozan Çatal, Johannes Nauta, Tim Verbelen, Pieter Simoons, and Bart Dhoedt. Bayesian policy selection using active inference. *ArXiv*, abs/1904.08149, 2019.
- [63] Beren Millidge. Deep active inference as variational policy gradients. *ArXiv*, abs/1907.03876, 2019.
- [64] Charles Blundell, Julien Cornebise, K. Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *ArXiv*, abs/1505.05424, 2015.
- [65] Jürgen Schmidhuber. Simple algorithmic principles of discovery, subjective beauty, selective attention, curiosity creativity. In *Discovery Science*, 2007.
- [66] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. 1991.
- [67] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.
- [68] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric P. Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *ArXiv*, abs/1906.05274, 2019.
- [69] Rein Houthoofd, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *NIPS*, 2016.
- [70] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 488–489, 2017.

- [71] Pranav Shyam, Wojciech Jaskowski, and Faustino Gomez. Model-based active exploration. In *ICML*, 2019.
- [72] Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *NIPS*, 2015.
- [73] Alexander Tschantz, Anil K. Seth, and Christopher L. Buckley. Learning action-oriented models through active inference. *bioRxiv*, 2019.
- [74] C. D. Freeman, Luke Metz, and David Ha. Learning to predict without looking ahead: World models without forward prediction. *ArXiv*, abs/1910.13038, 2019.
- [75] Alexander Tschantz, Beren Millidge, Anil K. Seth, and Christopher L. Buckley. Reinforcement learning through active inference. *ArXiv*, abs/2002.12636, 2020.
- [76] Beren Millidge, Alexander Tschantz, and Christopher L. Buckley. Whence the expected free energy? *ArXiv*, abs/2004.08128, 2020.
- [77] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2014.
- [78] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2016.
- [79] Samuel J. Gershman and Nathaniel D. Daw. Reinforcement learning and episodic memory in humans and animals: An integrative framework. *Annual Review of Psychology*, 68:101–128, 2017.
- [80] Nathaniel D. Daw, Yael Niv, and Peter Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8:1704–1711, 2005.
- [81] Endel Tulving. Episodic and semantic memory. 1972.
- [82] D. Schacter, D. Addis, and R. Buckner. Remembering the past to imagine the future: the prospective brain. *Nature Reviews Neuroscience*, 8:657–661, 2007.
- [83] D. Schacter, D. Addis, D. Hassabis, V. C. Martín, R. N. Spreng, and K. Szpunar. The future of memory: Remembering, imagining, and the brain. *Neuron*, 76:677–694, 2012.
- [84] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen. King, Dhharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

- [85] Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *ArXiv*, abs/1802.01561, 2018.
- [86] Steven Hansen, Pablo Sprechmann, Alexander Pritzel, Andr’e Barreto, and Charles Blundell. Fast deep reinforcement learning using online adjustments from the past. In *NeurIPS*, 2018.
- [87] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *CoRR*, abs/1511.05952, 2016.
- [88] Matthew M Botvinick, Sam Ritter, Jane X. Wang, Zeb Kurth-Nelson, and Demis Hassabis. Reinforcement learning, fast and slow. *Trends in cognitive sciences*, 23 5:408–422, 2019.
- [89] Andrea Klug. The hippocampus book. 2016.
- [90] Larry R. Squire. Memory systems of the brain: A brief history and current perspective. *Neurobiology of Learning and Memory*, 82:171–177, 2004.
- [91] David Marr. Simple memory: a theory for archicortex. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 262 841:23–81, 1971.
- [92] Robert J. Sutherland and Jerry W. Rudy. Configural association theory: The role of the hippocampal formation in learning. 1989.
- [93] Kenneth A. Norman and Randall C. O’Reilly. Modeling hippocampal and neocortical contributions to recognition memory: a complementary-learning-systems approach. *Psychological review*, 110 4:611–46, 2003.
- [94] Endel Tulving, Christopher Hayman, and Callum Macdonald. Long-lasting perceptual priming and semantic learning in amnesia: a case experiment. *Journal of experimental psychology. Learning, memory, and cognition*, 17 4:595–617, 1991.
- [95] Charles Blundell, Benigno Uria, Alexander Pritzel, Yazhe Li, Avraham Ruderman, Joel Z. Leibo, Jack W. Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *ArXiv*, abs/1606.04460, 2016.
- [96] Alexander Pritzel, Benigno Uria, Sriram Srinivasan, Adrià Puigdomènech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. *ArXiv*, abs/1703.01988, 2017.
- [97] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *ArXiv*, abs/1410.5401, 2014.
- [98] Junhyuk Oh, Valliappa Chockalingam, Satinder P. Singh, and Honglak Lee. Control of memory, active perception, and action in minecraft. *ArXiv*, abs/1605.09128, 2016.

- [99] Guangxiang Zhu, Zichuan Lin, Guangwen Yang, and Chongjie Zhang. Episodic reinforcement learning with associative memory. In *ICLR*, 2020.
- [100] Hyunwoo Jung, Moonsu Han, Minki Kang, and Sung Ju Hwang. Learning what to remember: Long-term episodic memory networks for learning from streaming data. *ArXiv*, abs/1812.04227, 2018.
- [101] Lukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to remember rare events. *ArXiv*, abs/1703.03129, 2017.
- [102] S. Klein, J. Loftus, and J. Kihlstrom. Memory and temporal experience: The effects of episodic memory loss on an amnesic patient’s ability to remember the past and imagine the future. *Social Cognition*, 20:353–379, 2002.
- [103] D. Hassabis, D. Kumaran, S. Vann, and E. Maguire. Patients with hippocampal amnesia cannot imagine new experiences. *Proceedings of the National Academy of Sciences*, 104:1726 – 1731, 2007.
- [104] Kourken Michaelian. Mental time travel: Episodic memory and our knowledge of the personal past. 2016.
- [105] Zafeirios Fountas, Anastasia Sylaidi, Kyriacos Nikiforou, Anil K. Seth, Murray Shanahan, and Warrick Roseboom. A predictive processing model of episodic memory and time perception. *bioRxiv*, 2020.
- [106] Warrick Roseboom, Z. Fountas, Kyriacos Nikiforou, David Bhowmik, M. Shanahan, and A. Seth. Activity in perceptual classification networks as a basis for human subjective time perception. *Nature Communications*, 10, 2019.
- [107] Andrea Greve, Elisa Cooper, Alexander Kaula, Michael C. Anderson, and Richard N. A. Henson. Does prediction error drive one-shot declarative learning? *Journal of Memory and Language*, 94:149 – 165, 2017.
- [108] Anthony I Jang, Matthew R. Nassar, Daniel G. Dillon, and Michael J. Frank. Positive reward prediction errors strengthen incidental memory encoding. *bioRxiv*, 2018.
- [109] Nina Rouhani, Kenneth A. Norman, and Yael Niv. Dissociable effects of surprising rewards on learning and memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 44:1430–1443, 2018.
- [110] Karl Pertsch, Oleh Rybkin, J. Yang, Shenghao Zhou, K. Derpanis, Kostas Daniilidis, Joseph J. Lim, and Andrew Jaegle. Keyframing the future: Keyframe discovery for visual prediction and planning. *arXiv: Learning*, 2020.
- [111] Cameron Browne, Edward Jack Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez Liebana, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4:1–43, 2012.

- [112] Marta Suárez-Pinilla, Kyriacos Nikiforou, Z. Fountas, A. Seth, and Warrick Roseboom. Perceptual content, not physiological signals, determines perceived duration when viewing dynamic, natural scenes. 2019.
- [113] Arash Vahdat, W. Macready, Zhengbing Bian, and Amir Khoshaman. Dvae++: Discrete variational autoencoders with overlapping transformations. In *ICML*, 2018.
- [114] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, A. Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *ArXiv*, abs/2006.15055, 2020.
- [115] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *ArXiv*, abs/1906.08253, 2019.
- [116] Christopher L. Buckley, Chang Sub Kim, Simon McGregor, and Anil K. Seth. The free energy principle for action and perception: A mathematical review. *Journal of Mathematical Psychology*, 81:55–79, 2017.
- [117] Karl J. Friston, Spyridon Samothrakis, and P. Read Montague. Active inference and agency: optimal control without cost functions. *Biological Cybernetics*, 106:523–541, 2012.
- [118] Philipp Schwartenbeck, Johannes Passecker, Tobias U. Hauser, Thomas HB Fitzgerald, Martin Kronbichler, and Karl J. Friston. Computational mechanisms of curiosity and goal-directed exploration. *eLife*, 8, 2019.
- [119] Martin Biehl, Christian Guckelsberger, Christoph Salge, Simón C. Smith, and Daniel Polani. Expanding the active inference landscape: More intrinsic motivations in the perception-action loop. *Frontiers in Neurobotics*, 12, 2018.
- [120] Maell Cullen, Ben Davey, Karl J. Friston, and Rosalyn J. Moran. Active inference in openai gym: A paradigm for computational investigations into psychiatric illness. *Biological psychiatry. Cognitive neuroscience and neuroimaging*, 3 9:809–818, 2018.
- [121] Andrea Banino, Adrià Puigdomènech Badia, Raphael Köster, Martin J. Chadwick, Vinícius Flores Zambaldi, Demis Hassabis, Caswell Barry, Matthew M Botvinick, Dharshan Kumaran, and Charles Blundell. Memo: A deep network for flexible combination of episodic memories. *ArXiv*, abs/2001.10913, 2020.
- [122] Demis Hassabis, Dharshan Kumaran, and Eleanor A. Maguire. Using imagination to understand the neural basis of episodic memory. *The Journal of neuroscience : the official journal of the Society for Neuroscience*, 27 52:14365–74, 2007.
- [123] Daniel L. Schacter, Donna Rose Addis, Demis Hassabis, V Cabezas Martín, R. Nathan Spreng, and Karl K Szpunar. The future of memory: Remembering, imagining, and the brain. *Neuron*, 76:677–694, 2012.

- [124] Demis Hassabis, Dhharshan Kumaran, Seralynne D. Vann, and Eleanor A. Maguire. Patients with hippocampal amnesia cannot imagine new experiences. *Proceedings of the National Academy of Sciences of the United States of America*, 104 5:1726–31, 2007.
- [125] Zichuan Lin, Tianqi Zhao, Guangwen Yang, and Lintao Zhang. Episodic memory deep q-networks. *ArXiv*, abs/1805.07603, 2018.
- [126] Pierre-Yves Oudeyer and Frédéric Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in Neurorobotics*, 1, 2009.
- [127] Prioritised Experience Replay in Deep Q Learning - Adventures in ML. <https://adventuresinmachinelearning.com/prioritised-experience-replay>.
- [128] P. Schwartenbeck, J. Passecker, T. Hauser, T. H. Fitzgerald, M. Kronbichler, and Karl J. Friston. Computational mechanisms of curiosity and goal-directed exploration. *eLife*, 8, 2019.
- [129] High-performance container datatypes. <https://docs.python.org/2/library/collections.html#collections.deque>.
- [130] Yarín Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*, 2016.
- [131] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [132] Benjamin Beyret, Jos’e Hern’andez-Orallo, Lucy Cheke, Marta Halina, Murray Shanahan, and Matthew Crosby. The animal-ai environment: Training and testing animal-like artificial cognition. 2019.

A Implementation of Prioritised Replay Buffer

Prioritised Replay Buffer (PRB) is used to *save* experienced transitions and *sample* them at training proportional to a chosen metric (Section 4). Our implementation of PRB consists of three main components:

1. *SimpleMemory* class that stores all the information about a single transition needed for training the networks.
2. *Data structure to store* a large number of SimpleMemory objects (100,000) and to access them efficiently at sampling.
3. *Efficient sampling algorithm* that allows sampling the transitions in $O(\log N)$.

The SimpleMemory class is a weak form of episodic memory that only stores information about a single given transition and, thus, does not incorporate any information about temporal dependencies within an episode. As such, SimpleMemory has the following attributes: i) preceding observation (o_0), ii) action taken after observing o_0 (a_0), iii) following observation (o_1), iv) individual free energies produced by each network at the latest training iteration with this transition (F_h, F_t, F_a)¹³. The values of individual free energies are used for priority sampling, depending on the sampling mode chosen.

To store these objects in an efficient manner, we use a collections.deque structure, which allows for memory-efficient appends and pops in $O(1)$, as opposed to $O(n)$ for a normal list [129]. We also specify the maximum length of the deque at 100,000.

To sample from a deque of 100,000 transitions, we implement a fast sampling method with the use of a data structure called sum-tree [78]. Sum-tree is a binary tree structure, where the values of the children of a node sum to the value of a node, see Figure 31. The values of the leaf nodes correspond to the corresponding metric chosen (e.g. free energy values of the transition model); hence, the top node is a sum of all free energy values in the buffer, F_{sum} .

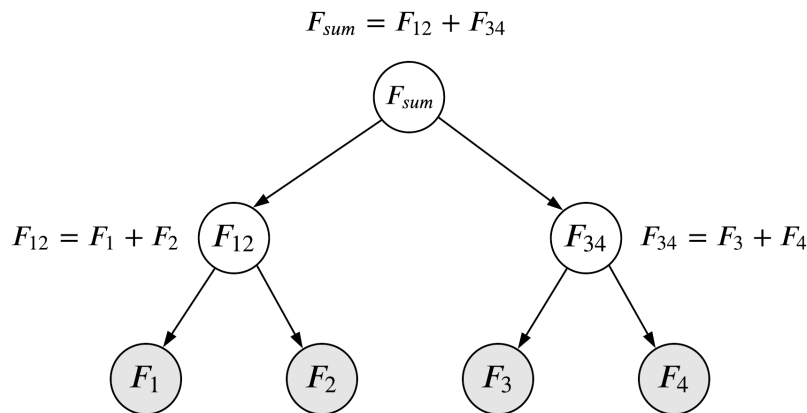


Figure 31: **Sum-tree structure.** The leaf nodes represent the free energy values of each transition in a buffer. The value of each parent node is a sum of its children.

¹³ h refers to the habitual network, t refers to the transition model, and a refers to the autoencoder

Sampling from this data structure is done by first sampling from a uniform distribution $v \sim U(0, F_{sum})$. Then, the structure is traversed downwards, in an iterative way, by first considering the left-hand child and comparing whether the sampled value, v , is less than the child's value. Once this condition is met, the algorithm moves to the right-hand child of the node where the condition has been met, and subtracts the left-hand child's value from v (such that in the next steps $v = v - \text{child}_{\text{left}}$). It then continues in the same iterative fashion until the bottom of the tree. When the algorithm reaches a leaf node, it retrieves its index (i.e. its location in the buffer), accesses it in the buffer, and places it in the next training batch – Figure 34. Our implementation was also inspired by [127].

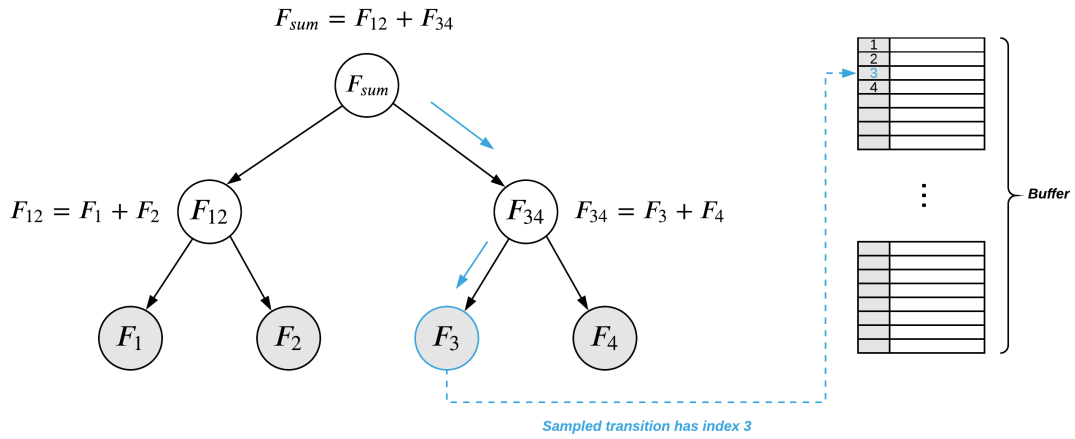


Figure 32: **Simplified example of sampling in the sum-tree structure.** Iterative sampling algorithm path (blue) reaches a leaf node, which refers the algorithm to a slot in the buffer.

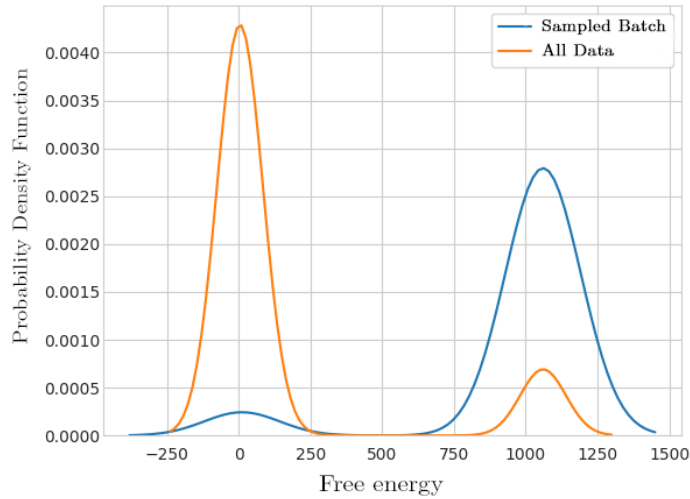


Figure 33: **Resulting sampled batch distribution** shifts towards higher values of free energy. *Orange:* probability density function (pdf) over the entire dataset; *Blue:* pdf over a single sampled batch with prioritisation.

B Additional results of On-line vs. PRB comparison

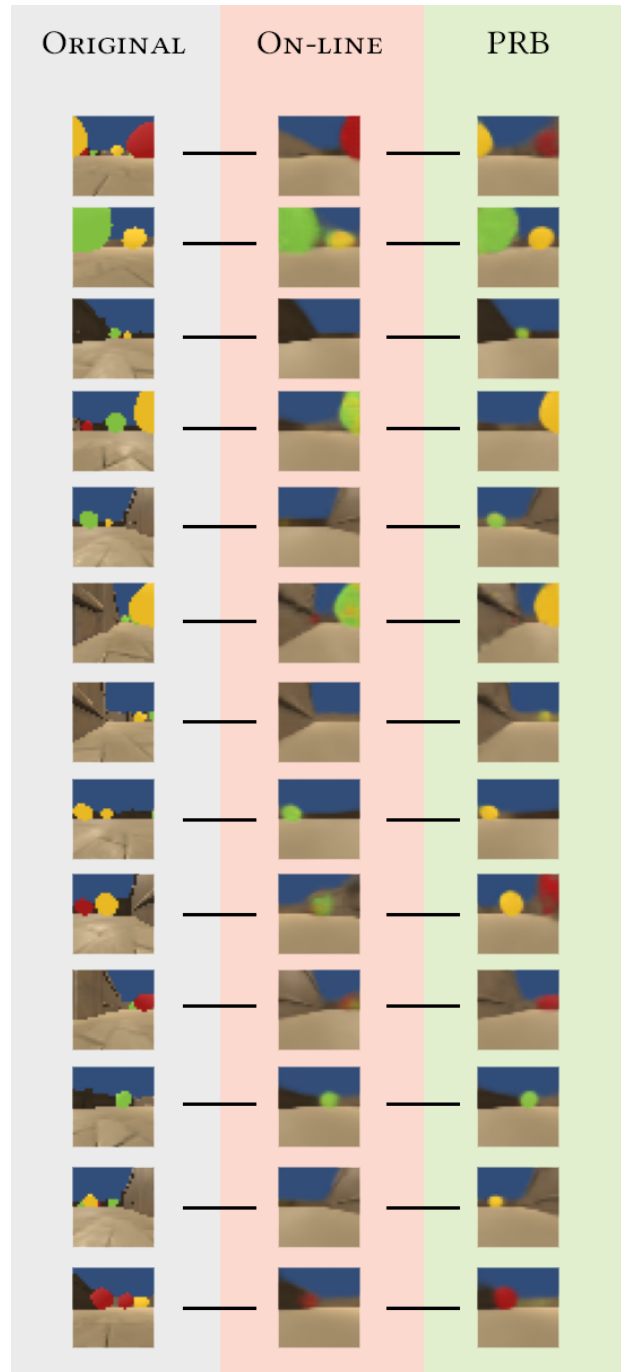


Figure 34: **Reconstruction Comparison of On-line vs. PRB agents.** These results showcase the improved performance of the autoencoder that was trained with PRB vs. the on-line approach. *Original* observations (on the left) were passed through the compared autoencoders (columns on the right).

C Architectural Details

C.1 On-line and PRB Agent

The on-line and PRB agents shared the same baseline architecture and training hyperparameters. As mentioned, each component of the generative and inference models is parametrised by feed-forward neural networks (including fully-connected, convolutional and transpose-convolutional layers), whose architectural details can be found in Figure 35. The latent bottleneck of the autoencoder, s , was made to be of size 10. The hyperparameters of the top-down attention mechanism were: $a = 2$, $b = 0.5$, $c = 0.1$, and $d = 5$. Further, we restricted the action space to just 3 actions – forward, left, right. For testing, we optimised the MCTS parameters of the PRB-MCTS agent, setting $c_{\text{explore}} = 0.1$ and $T_{\text{dec}} = 0.8$. Furthermore, we perform 30 simulation loops, each with depth of 1. The networks were trained using separate optimisers for stability reasons. The habitual and transition networks are trained with a learning rate of 0.0001; the autoencoder’s optimiser had a learning rate of 0.001. Both on-line and PRB agents were trained on batch size 50. All of the networks were implemented using Tensorflow v2.2 [131]. Tests were performed in Animal-AI v2.0.1 [11].

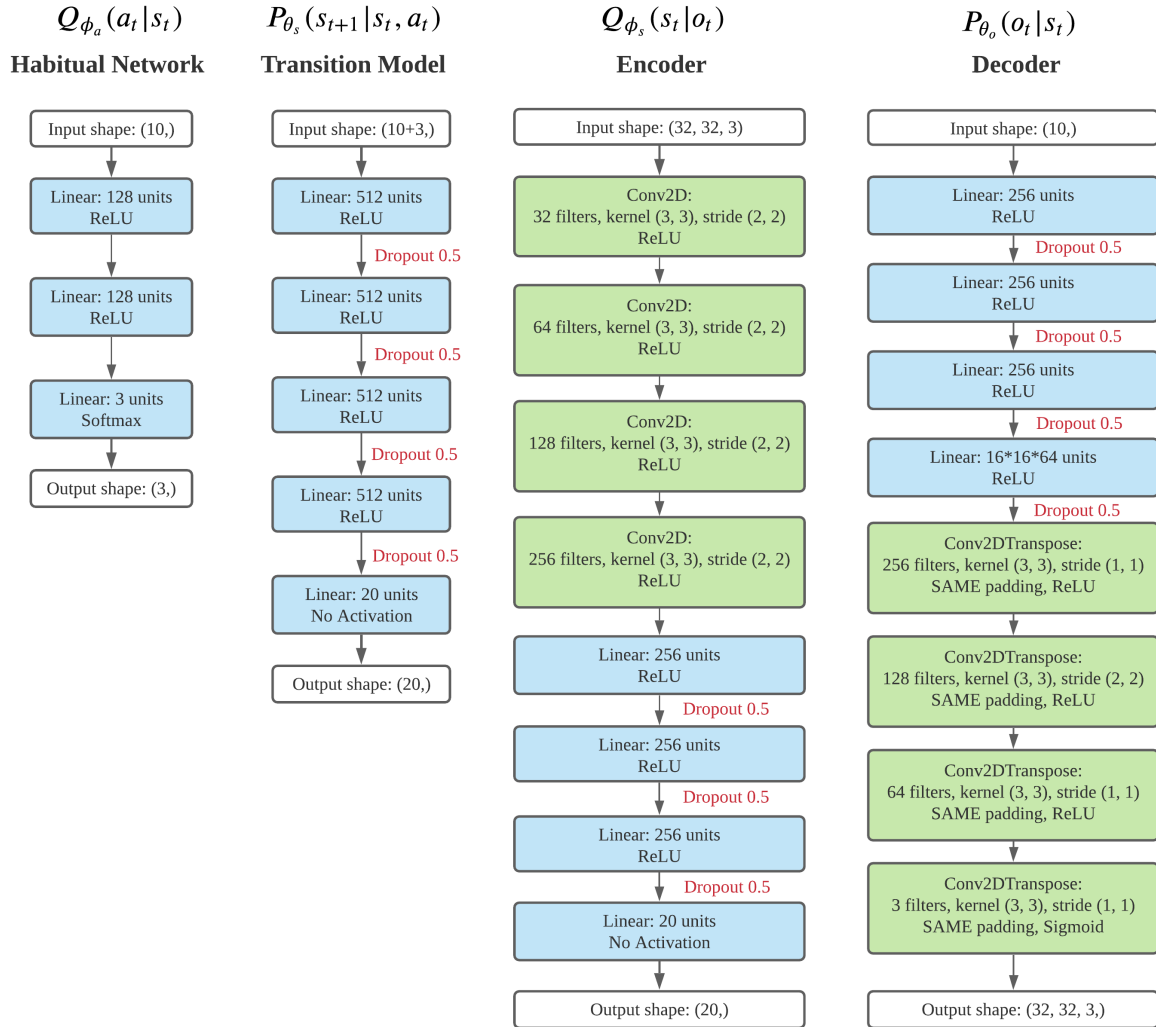


Figure 35: Baseline agent architecture.

As discussed, each network was trained with its corresponding loss function, which are the constituent parts of the total variational free energy. In particular, autoencoder was trained using Eqs. 26a and 26b, transition model using Eq. 26b, and habitual network using Eq. 26c.

Furthermore, following the training procedure from Fountas et al. [10], we stabilise the convergence of the autoencoder by modifying the loss function to:

$$\mathcal{L}_{\text{autoencoder}} = -\mathbb{E}_{Q(s_t)} \left[\log P_{\theta_o}(o_t|s_t) \right] + \gamma D_{\text{KL}} \left[Q_{\phi_s}(s_t) || P_{\theta_s}(s_t|s_{t-1}, a_{t-1}) \right] \quad (39)$$

$$+ (1 - \gamma) D_{\text{KL}} \left[Q_{\phi_s}(s_t) || N(0, 1) \right], \quad (40)$$

where γ is a hyperparameter that gradually increases from 0 to 0.8 during training.

C.2 TEM Architecture

The TEM transition model was trained on batch size 15 and a learning rate of 0.0005. Each batch consisted of zero-padded sequences with length 50. Further, we use a Masking layer to ignore zero-padded parts of the sequences in the computational graph. The training was stopped at 200k training iterations. For testing TEM-MCTS, we optimise the MCTS parameters, setting $c_{explore} = 0.1$ and $T_{dec} = 0.8$. Furthermore, we perform 15 simulation loops, each with depth of 3.

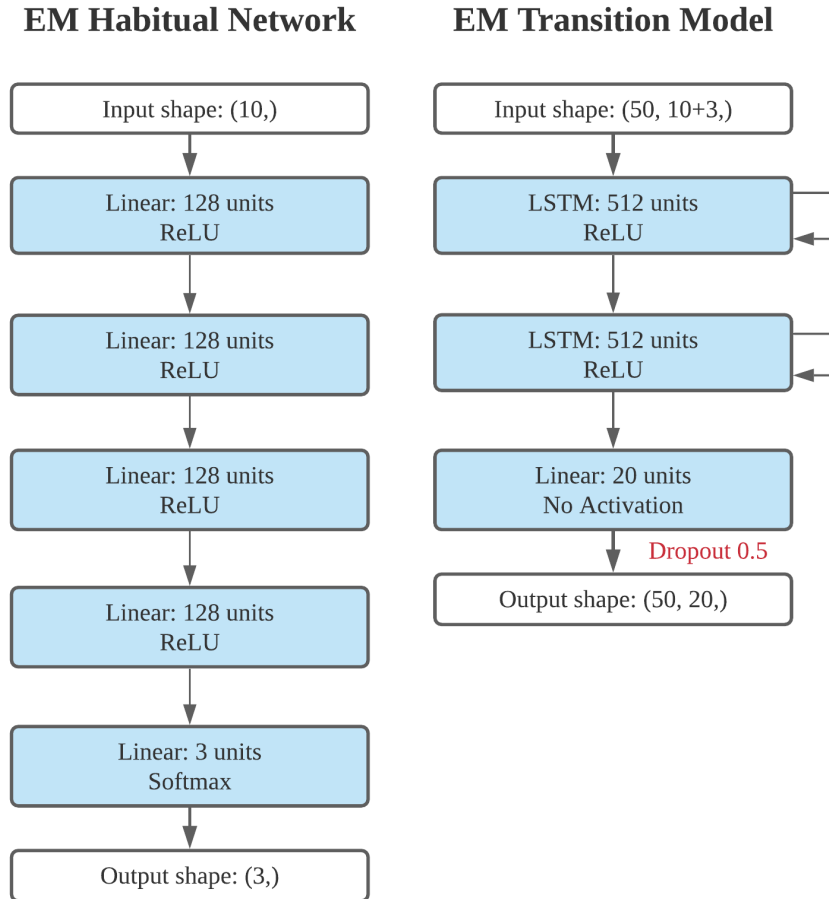


Figure 36: Additional networks introduced by the TEM module.

D PRB Transition Model Imagination

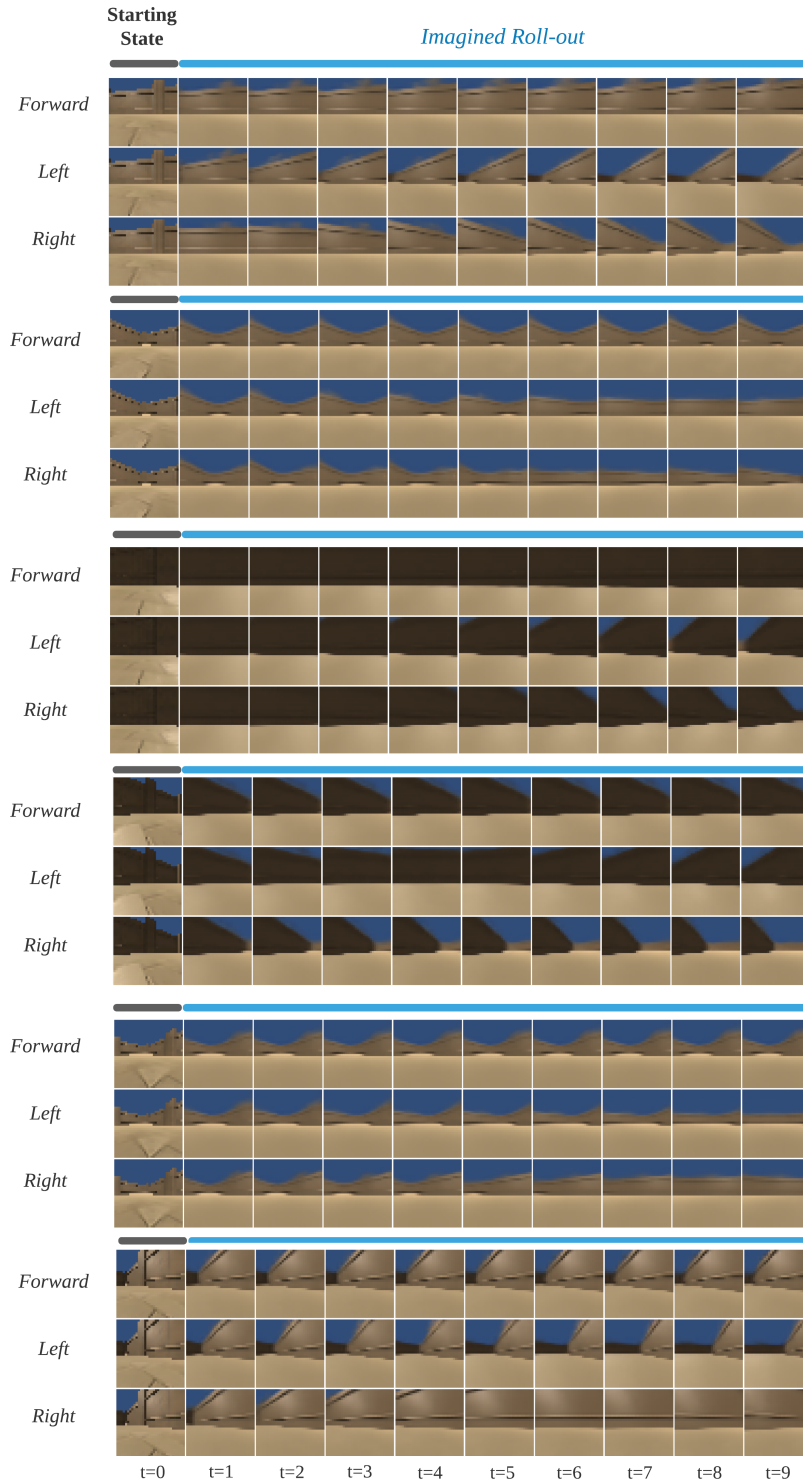


Figure 37: **The original, slow transition model is not able to imagine objects.** As discussed in the main body of the report, the slow transition model is not able to imagine objects appearing in the frame as a direct corollary of how it is trained.