Deconstructing deep active inference.

Théophile Champion TMAC3@KENT.AC.UK
University of Kent, School of Computing
Canterbury CT2 7NZ, United Kingdom

Marek Grześ M.GRZES@KENT.AC.UK
University of Kent, School of Computing
Canterbury CT2 7NZ, United Kingdom

Lisa Bonheme LB732@KENT.AC.UK
University of Kent, School of Computing
Canterbury CT2 7NZ, United Kingdom

Howard Bowman H.BOWMAN@KENT.AC.UK
University of Birmingham, School of Psychology,
Birmingham B15 2TT, United Kingdom
University of Kent, School of Computing
Canterbury CT2 7NZ, United Kingdom
University College London, Wellcome Centre for Human Neuroimaging
London WC1N 3AR, United Kingdom

Editor: TO BE FILLED

Abstract

Active inference is a theory of perception, learning and decision making, which can be applied to neuroscience, robotics, psychology, and machine learning. Recently, intensive reasearch has been taking place to scale up this framework using Monte-Carlo tree search and deep learning. The end-goal of this activity is to solve more complicated tasks using deep active inference. First, we review the existing literature, then, we progresively build a deep active inference agent as follows: (i) implement a variational auto-encoder (VAE), (ii) implement a deep hidden Markov model (HMM), (iii) implement a deep critical hidden Markov model (CHMM), and (iv) implement a complete deep active inference agent (DAI). For the CHMM and DAI ~~agent~~agents, we have experimented with five definitions of the expected free energy and three different action selection strategies. According to our experiments, the models able to solve the dSprites environment are the ones that maximise rewards. Finally, we compare the similarity of the representation learned by the layers of various models (e.g., deep Q-network, CHMM, DAI~~, etc...~~) using centered kernel alignment. Importantly, the CHMM maximising reward and the CHMM minimising expected free energy learn very similar representations except for the last layer of the critic network (reflecting the difference in learning objective), and the variance layers of the transition and encoder networks. While performing further inspection of those (variance) layers, we found that the transition network of the reward maximising CHMM is a lot more certain than the transition network of the CHMM minimising expected free energy. More precisely, the CHMM minimising expected free energy is only confident about the world transition when performing action down. This suggests that the CHMM minimising expected free energy always picks the action down, and does not gather enough data for the other actions. In contrast, the CHMM maximising reward, keeps on selecting the actions left and right, enabling it to successfully solve the task. The only difference between those two CHMMs is the epistemic value, which aims to ~~makes~~ make the outputs of the transition and encoder networks as close as possible. Thus, the CHMM minimising expected free energy repeatedly picks a single action (down), and becomes an expert at predicting the future when selecting this action. This effectively makes the KL divergence between the output of the transition and encoder networks small. Additionally, when selecting the action down the average reward is zero, while for all the other actions, the expected reward will be negative. Therefore, if the CHMM has to stick to a single action to keep the KL divergence small, then the action down is the most rewarding. Thus, the appropriate formulation of the epistemic value in deep active inference remains an open question.

Keywords: Deep Learning, Active Inference, Bayesian Statistics, Free Energy Principle, Reinforcement Learning

1. Introduction

Active inference is a unified framework for perception, learning, and planning that has emerged from theoretical neuroscience (Costa et al., 2020a; Champion et al., 2021c,b,a,d). This framework has successfully explained a wide range of brain phenomena (Friston et al., 2016; Itti and Baldi, 2009; Schwartenbeck et al., 2018; FitzGerald et al., 2015), and has been applied to a large number of tasks in robotics and artificial intelligence (Fountas et al., 2020; Pezzato et al., 2020; Sancaktar et al., 2020; Çatal et al., 2020; Cullen et al., 2018; Millidge, 2019).

A promising area of research revolves around scaling up this theoretical framework to tackle increasingly complex tasks. Research towards this goal is generally driven from recent advances in machine learning. For example, variational auto-encoders ~~(Doersch, 2016; Higgins et al., 2017)~~ (Doersch, 2016; Higgins et al., 2017; Kingma and Welling, 2014; have been key to the integration of deep neural networks within active inference (Sancaktar et al., 2020; Çatal et al., 2020; Millidge, 2020), and the Monte Carlo tree search algorithm ~~(?Silver et al., 2016)~~ (Browne et al., 2012; Silver et al., 201 has been used to improve planning efficency (Fountas et al., 2020; Champion et al., 2021b,a,d, 2022).

Another closely related field is reinforcement learning (Mnih et al., 2013; van Hasselt et al., 2015; Lample and Chaplot, 2016), which addresses the same kind of tasks, where an agent must interact with its environment. A known challenge in this field is the correlation between the consecutive samples, which violates the standard i.i.d. assumption on which most of machine learning relies. To break this correlation, researchers proposed to store past experiences of the agent inside a replay buffer (Mnih et al., 2013). Experiences can then be re-sampled randomly from the buffer to train the Q-network, which is used to approximate Q-values. The Q-network is trained to minimize the mean squared error between its output and a target value, which is defined as:

$$y(o_t, a_t) = \mathbb{E}_{o_{t+1} \sim E(o_t, a_t)}\Big[r_t + \gamma \max_{a_{t+1} \in \mathcal{A}} \mathcal{Q}_{\theta_a}(o_{t+1}, a_{t+1})\Big],$$

where t is the present time step, $\mathcal{A}$ is the set of available actions, $y(o_t, a_t)$ is the target Q-value to be predicted, $\mathbb{E}$ is the expectation w.r.t the observations received from the environment, r_t is the reward obtained by the agent when performing action a_t in state[1] o_t, o_{t+1} is the state reached when performing action a_t in state o_t, E is the environment emulator from which o_{t+1} is sampled, γ is the discount factor that discounts future rewards, and $\mathcal{Q}_{\theta_a}(o_{t+1}, a_{t+1})$ is the output of the Q-network, i.e., the estimated Q-value of performing action a_{t+1} in state o_{t+1}.

Unfortunatly, using the above target to train the Q-network can make the training unstable. Generally, the problem is addressed by introducing a target network $\hat{\mathcal{Q}}_{\hat{\theta}_a}(o_{t+1}, a_{t+1})$, which is simply a copy of the Q-network. The weights of the target network are then synchronized with the weights of the Q-network every K (learning) iterations (Mnih et al., 2013). The new target is obtained by replacing the Q-network by the target network, i.e.,

$$y(o_t, a_t) = \mathbb{E}_{o_{t+1} \sim E(o_t, a_t)}\Big[r_t + \gamma \max_{a_{t+1} \in \mathcal{A}} \hat{\mathcal{Q}}_{\hat{\theta}_a}(o_{t+1}, a_{t+1})\Big].$$

In Section 2, we review the existing literature and present: the Deep Q-network (DQN) agent (Mnih et al., 2013), the deep active inference with Monte-Carlo methods (DAI_{MC}) agent by Fountas et al. (2020), the deep active inference as variational policy gradients (DAI_{VPG}) approach by Millidge (2020), the deep active inference agent of rubber hand illusion (DAI_{RHI}) by Rood et al. (2020), the deep active inference agent for humanoid robot control (DAI_{HR}) by Sancaktar et al. (2020); Lanillos et al. (2020); Oliver et al. (2019), the deep active inference agent based on the free action objective (DAI_{FA}) by Ueltzhöffer (2018), a deep active inference agent for partially observable Markov decision processes (DAI_{POMDP}) by van der Himst and Lanillos (2020), as well as various methods for which the code is not available online. We argue that while all these approaches illuminate important issues associated with realising a deep active inference agent, a fully complete implementation has not yet been published. Consequently, to systematically explore the construction of deep active inference agents, in Section 3, we incrementally build such an agent. We start with a simple variational auto-encoder (VAE) composed of an encoder and decoder network. Next, a transition network is added to create a deep hidden Markov model (HMM). Then, a critic network is added to define a prior over actions, which leads to the critical HMM (CHMM). Lastly, the policy network is added to approximate the posterior over actions leading to the full deep active inference (DAI) agent. Then, in Section 4, we discuss our

[1] Note, we are using the notation o_τ for the (observable) state at (an arbitrary) time step τ, instead of the more standard notation s_τ. This is because we reserve the notation s_τ for the (unobserved) states that arise in the context of active inference.

findings regarding the abilities and limitations of each intermediate step. This section also presents an analysis and discussion of the representations learned by each intermediate model. Finally, Section 5 puts our findings in context and concludes this paper.

2. Review of existing research

In this section, we discuss the DQN agent from the reinforcement learning literature, six agents from the active inference literature for which the code is available online (DAI_{MC}, DAI_{VPG}, DAI_{RHI}, DAI_{HR}, DAI_{FA}, and DAI_{POMDP}), and a few other deep active inference agents for which the code is unavailable. Finally, we explain how the representations learned by the agents can be compared using centered kernel alignment. Note, the notation used throughout this section is summarised in Appendix A.

2.1 DQN agent (Mnih et al., 2013)

Let us start with the DQN agent (Mnih et al., 2013), whose goal is to maximise the amount of reward obtained over time. At each time step τ, the agent is observing an image o_τ, and is allowed to perform one action $a_\tau \in \mathcal{A}$. After performing a_τ when observing o_τ, the agent receives a reward r_τ. The Q-learning algorithm (Sutton et al., 1998) aims to maximise reward by computing the Q-values $Q(o_\tau, a_\tau)$, for each state-action pair (o_τ, a_τ). The Q-values represent the expected amount of rewards obtained by taking action a_τ in state o_τ. This approach is intractable for image based domains such as Atari games, since one would need to store a vector of Q-values for each possible image. Instead, the DQN algorithm (illustrated in Figure 1) has been developed, which uses a deep neural network $\mathcal{Q}_{\theta_a}$ to approximate the Q-values. More formally, $\mathcal{Q}_{\theta_a}$ maps any observation to a vector of size $\#A$ containing the Q-values of each possible action, and we denote by $\mathcal{Q}_{\theta_a}(o_\tau, a_\tau)$ the element at position a_τ in the output vector predicted by Q_{θ_a} when provided with the image o_τ. As we discussed in the introduction, the training stability of the Q-network is improved by introducing a target network $\hat{\mathcal{Q}}_{\hat{\theta}_a}$, which is structurally identical to the Q-network and whose weights are synchronised with the weights of the Q-network every K (learning) iterations. The Q-network's weights are then optimised using gradient descent to minimise the mean square error between the output of the Q-network and a target value, i.e., $\theta_a^* = \arg\min_{\theta_a} \text{MSE}[\mathcal{Q}_{\theta_a}(o_t, \bullet), y(o_t, \bullet)]$, where $y(o_t, a_t)$ is the target Q-value for each state-action pair, and, as highlighted earlier, is defined as follows:

$$y(o_t, a_t) = \mathbb{E}_{o_{t+1} \sim E(o_t, a_t)} \Big[r_t + \gamma \max_{a_{t+1} \in \mathcal{A}} \hat{\mathcal{Q}}_{\hat{\theta}_a}(o_{t+1}, a_{t+1}) \Big].$$

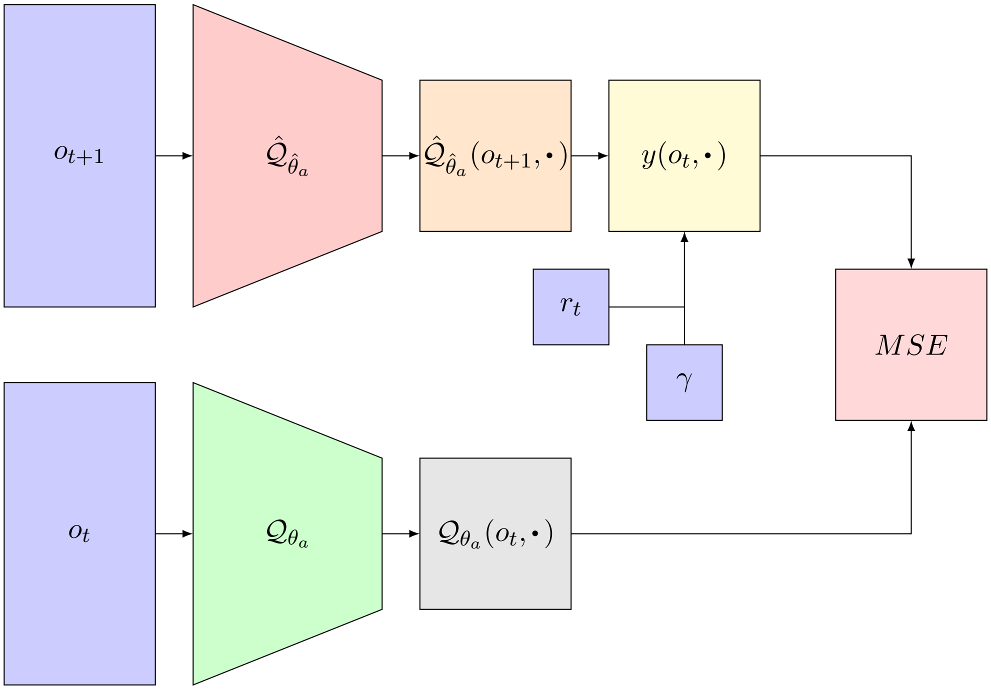

o_{t+1}
$\hat{\mathcal{Q}}_{\hat{\theta}_a}$
$\hat{\mathcal{Q}}_{\hat{\theta}_a}(o_{t+1},\bullet)$
$y(o_t,\bullet)$
r_t
γ
MSE
o_t
$\mathcal{Q}_{\theta_a}$
$\mathcal{Q}_{\theta_a}(o_t,\bullet)$

Figure 1: This figure illustrates the DQN agent. Briefly, the image o_t is fed into the Q-network, and the image o_{t+1} is fed into the target network. The Q-network outputs the Q-values for each action at time t, and the target network outputs the Q-values for each action at time $t+1$. Then, the reward, the discount factor, and Q-values of each action at time $t+1$ are used to compute the target values $y(o_t,\bullet)$. Finally, the goal is to minimise the MSE between the prediction of the Q-network and the target values by changing the weights of the Q-network.

2.2 DAI_{MC} agent (Fountas et al., 2020)

In this section, we review the DAI_{MC} agent proposed by Fountas et al. (2020). The relevant code is available at the following URL: `https://github.com/zfountas/deep-active-inference-mc`. The DAI_{MC} agent is composed of four deep neural networks illustrated in Figures 2 and 3. The encoder $\mathcal{E}_{\phi_s}$ takes images as input, and outputs the mean and variance of the variational distribution over hidden states, i.e., $Q_{\phi_s}(s_t) = \mathcal{N}(s_t; \mu, \sigma)$, where $\mu, \sigma = \mathcal{E}_{\phi_s}(o_t)$. The decoder $\mathcal{D}_{\theta_o}$ takes a state as input, and outputs the parameters of a product of Bernoulli distributions, which can be interpreted as the expected (reconstructed) image $\hat{o}_t$, i.e.,

$$P_{\theta_o}(o_t|s_t) = \mathcal{B}\text{ernoulli}(o_t; \hat{o}_t),$$

where $\hat{o}_t = \mathcal{D}_{\theta_o}(s_t)$ are the values predicted by the decoder, and $\mathcal{B}\text{ernoulli}(o_t; \hat{o}_t)$ is a product of Bernoulli distributions defined as:

$$\mathcal{B}\text{ernoulli}(o_t; \hat{o}_t) = \prod_{x,y} \text{Bernoulli}(o_t[x,y]; \hat{o}_t[x,y]),$$

where Bernoulli$(\bullet;\bullet)$ is a Bernoulli distribution over the possible values of the pixel $o_t[x,y]$, parameterized by the parameter $\hat{o}_t[x,y]$. The transition network $\mathcal{T}_{\theta_s}$ takes a state-action pair as input, and outputs the mean and variance of a Gaussian distribution over hidden states, i.e., $P_{\theta_s}(s_{\tau+1}|s_\tau, a_\tau) = \mathcal{N}(s_{\tau+1}; \mathring{\mu}, \frac{\mathring{\sigma}}{\omega_t})$, where $\mathring{\mu}, \mathring{\sigma} = \mathcal{T}_{\theta_s}(s_\tau, a_\tau)$, and ω_t is the top-down attention parameter modulating the precision of the transition mapping (see below). The policy network $\mathcal{P}_{\phi_a}$ takes a state as input, and outputs a distribution over actions, i.e., $Q_{\phi_a}(a_t|s_t) = \text{Cat}(a_t; \hat{\pi})$, where $\hat{\pi} = \mathcal{P}_{\phi_a}(s_t)$. Finally, the prior over actions is defined as follows:

$$P(a_t) = \sum_{\pi \in \Pi} [\pi_t = a_t] P(\pi), \tag{1}$$

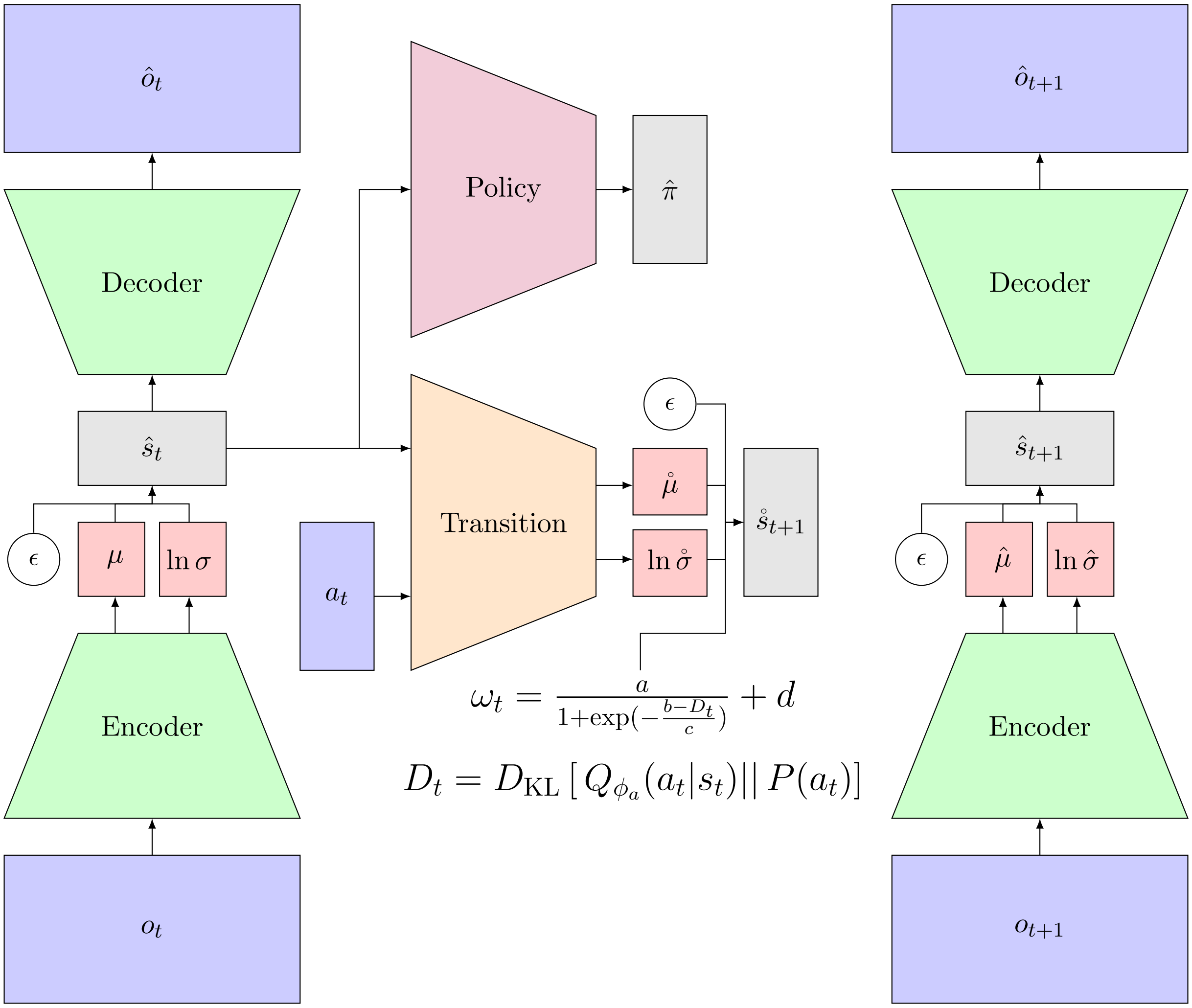


Figure 2: This figure illustrates the DAI_{MC} agent, which is composed of an encoder, a decoder, a transition network, and a policy network. The same VAE (encoder and decoder) is repeated in the figure to reflect successive time-points.

where Π is the set of all possible policies, π_t is the action precribed by policy π at time t, the square brackets represent an indicator function that equals one if the condition within the bracket is satisfied and zero otherwise, and $P(\pi)$ is the prior over policies defined as:

$$P(\pi) = \sigma[-G(\pi)],$$

where $\sigma[\bullet]$ is the softmax function, and $G(\pi)$ is the expected free energy (EFE) of policy π, which is defined as:

$$G(\pi) = \sum_{\tau=t}^{T} G_\tau(\pi) = \sum_{\tau=t}^{T} \mathbb{E}_{\tilde{Q}}\Big[\ln Q(s_\tau, \theta|\pi) - \ln \tilde{P}(o_\tau, s_\tau, \theta|\pi)\Big], \tag{2}$$

where $\tilde{Q} = Q(o_\tau, s_\tau, \theta|\pi) = Q(\theta|\pi)Q(s_\tau|\theta, \pi)Q(o_\tau|s_\tau, \theta, \pi)$ and $\tilde{P}(o_\tau, s_\tau, \theta|\pi) = P(o_\tau|\pi)P(s_\tau|o_\tau, \pi)P(\theta|s_\tau, o_\tau, \pi)$. However, ~~this is not the form of the expected free energy that is computed in the Fountas et al. (2020) approach. We will shortly present derivations that lead to the terms that are computed. Up to now, we left the definition of~~ Equation 2 needs to be re-arranged to be computed in practice[2], and Section 2.2.3 will present this derivation. Finally, as shown in Figure 2, the top-down attention parameter ~~undefined. Formally, this parameter~~ is computed as

[2] By “in practice”, we mean “in the code” or equivalently “when implementing the approach”.

follows:

$$\omega_t = \frac{a}{1 + \exp(-\frac{b - D_t}{c})} + d,$$

where $D_t = D_{\text{KL}}\left[\, Q_{\phi_a}(a_t|s_t) || \, P(a_t)\right]$, and $\{a, b, c, d\}$ are fixed hyperparameters. Intuitively, ω_t is high when the posterior over actions (from the policy network) is close to the prior over actions (from the expected free energy), and low when the posterior is far away from the prior. This, in turn, means that extra uncertainty is introduced into the transition mapping (see paragraph before Equation 1) when posterior over actions and prior over actions are very different. Finally, note that the number of terms required to compute the prior over actions (defined in Equation 1) grows exponentially with the time horizon of planning. Because this is intractable in practice, Fountas et al. (2020) implemented a Monte-Carlo tree search (MCTS) algorithm to evaluate the expected free energy of each action (see below). Finally, action selection is performed by sampling from the following distribution:

$$\tilde{P}(a_t) = \frac{N(\hat{s}_t, a_t)}{\sum_{\hat{a}_t} N(\hat{s}_t, \hat{a}_t)},$$

where $\hat{s}_t$ is the current state of the environment, and $N(s_t, a_t)$ is the number of times action a_t has been visited from state s_t during MCTS.

2.2.1 The Monte-Carlo tree search

In this section, we describe the planning algorithm used by DAI_{MC}, i.e., Monte-Carlo tree search (MCTS). MCTS is used to enhance the planning ability of the agent by allowing it to look into the future. At the beginning of an action-perception cycle, the agent is provided with an image o_t. This image can be feed into the encoder to get the mean vector μ of the posterior over the latent states, i.e., $Q_{\phi_s}(s_t) = \mathcal{N}(s_t; \mu, \sigma)$. Since μ is the mean of the Gaussian posterior, it can be interpreted as the maximum a posteriori (MAP) estimate of the latent states at time step t. This MAP estimate will constitute the root node of the Monte-Carlo tree search (MCTS).

The first step of the MCTS is to use the ~~UCT~~ Upper Confidence bounds for Trees (UCT) criterion to determine which node in the tree should be expanded. Let the tree's root $\hat{s}_t$ be called the current node, which is denoted $\hat{s}_\tau$. If the current node has no children (i.e., no previously selected actions from the current node), then it is selected for expansion. Alternatively, the child with the highest UCT criterion becomes the new current node and the process is iterated until we reach a leaf node (i.e. a node from which no action has previously been selected). The UCT criterion (Browne et al., 2012) of the child of $\hat{s}_\tau$ corresponding to action $\hat{a}_\tau$ is given by:

$$UCT(\hat{s}_\tau, \hat{a}_\tau) = -\bar{G}(\hat{s}_\tau, \hat{a}_\tau) + C_{explore} \cdot \frac{Q_{\phi_a}(a_\tau = \hat{a}_\tau | s_\tau = \hat{s}_\tau)}{1 + N(\hat{s}_\tau, \hat{a}_\tau)},$$

where $\bar{G}(\hat{s}_\tau, \hat{a}_\tau)$ is the average expected free energy of taking action $\hat{a}_\tau$ in state $\hat{s}_\tau$, $C_{explore}$ is the exploration constant that modulates the amount of exploration at the tree level, $N(\hat{s}_\tau, \hat{a}_\tau)$ is the number of times action $\hat{a}_\tau$ was visited in state $\hat{s}_\tau$, and $Q_{\phi_a}(a_\tau = \hat{a}_\tau | s_\tau = \hat{s}_\tau)$ is the posterior probability of action $\hat{a}_\tau$ in state $\hat{s}_\tau$ as predicted by the policy network.

Let $\mathring{s}_\tau$ be the (leaf) node selected by the above selection procedure. The MCTS then expands one of the children of $\mathring{s}_\tau$. The expansion uses the transition network to compute the mean $\mathring{\mu}$ of $P_{\theta_s}(s_{\tau+1} | s_\tau = \mathring{s}_\tau, a_\tau = \mathring{a}_\tau)$, which is viewed as a MAP estimate of the states at time $\tau + 1$. Then, we need to estimate the cost of (virtually) taking action $\mathring{a}_\tau$. By definition, the cost is the expected free energy given by (2), and Monte-Carlo rollouts can be run to improve its estimation. The final step of the planning iteration is to back-propagate the cost of the newly expanded (virtual) action toward the root of the tree. Formally, we write the update as follows:

$$\forall s \in \mathbb{A}_{\mathring{s}_\tau} \cup \{\mathring{s}_\tau\}, \quad \boldsymbol{G}_s \leftarrow \boldsymbol{G}_s + \boldsymbol{G}_{\mathring{s}_\tau}, \tag{3}$$

where $\mathring{s}_\tau$ is the node that was selected for expansion, $\boldsymbol{G}_s$ is the expected free energy of s, and $\mathbb{A}_{\mathring{s}_\tau}$ is the set of all ancestors of $\mathring{s}_\tau$ in the tree. During the back propagation, we also update the number of visits as follows:

$$\forall s \in \mathbb{A}_{\mathring{s}_\tau} \cup \{\mathring{s}_\tau\}, \quad \boldsymbol{N}_s \leftarrow \boldsymbol{N}_s + 1,1. \tag{4}$$

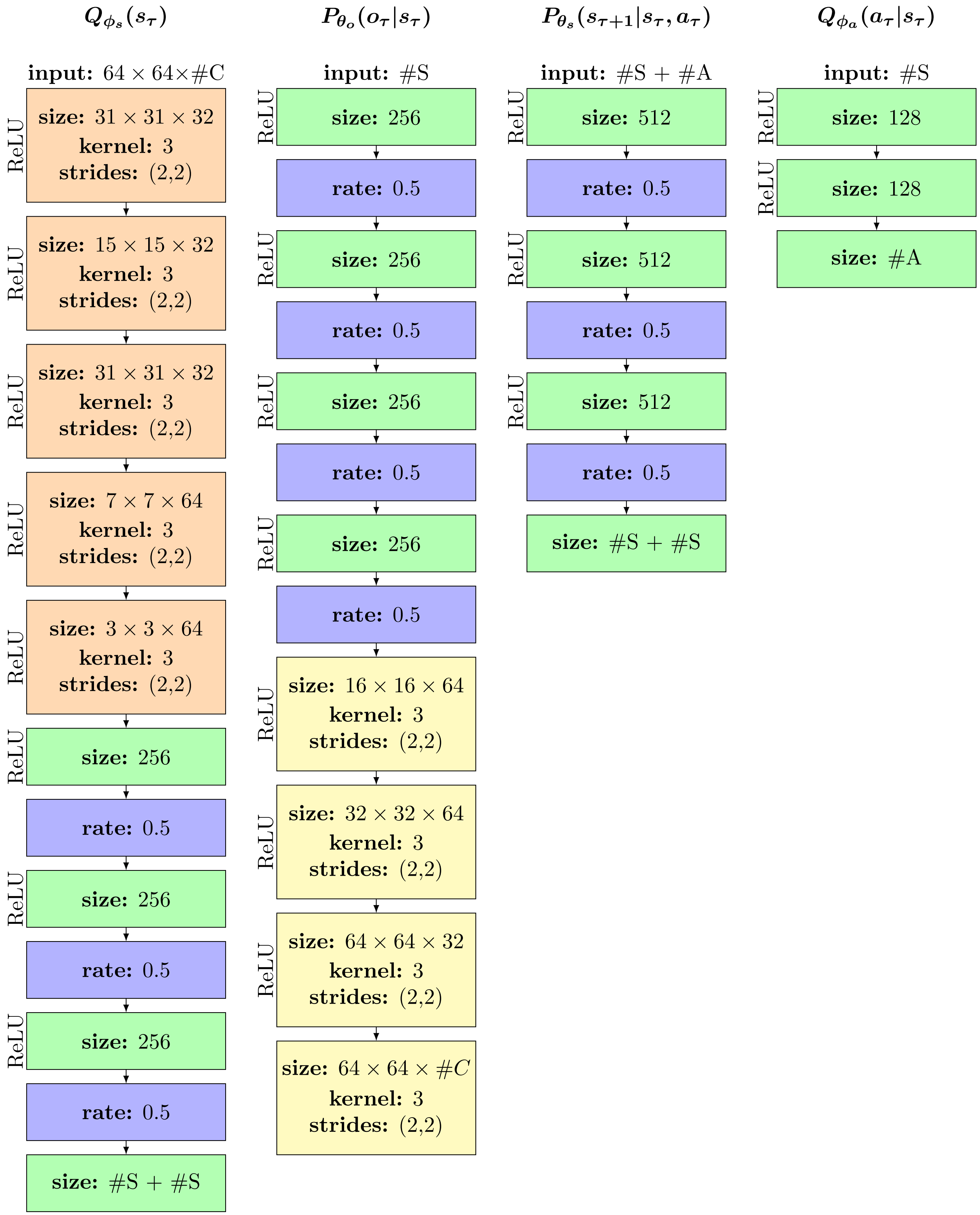


Figure 3: Neural network architectures of the DAI_{MC} agent. Orange blocks correspond to convolutional layers, green blocks correspond to fully connected layers, blue blocks correspond to dropout, and yellow blocks correspond to up-convolutional layers. For the dSprites environment, there are four actions (i.e., #A = 4), ten states (i.e., #S = 10), and only one channel (i.e., #C = 1). For the Animal-AI environment, there are three actions (i.e., #A = 3), ten states (i.e., #S = 10), and three channels (i.e., #C = 3). These are all trained to minimize variational free energy.

If we let $\boldsymbol{G}_s^{aggr}$ be the aggregated cost of an arbitrary node s obtained by applying Equation 3 after each expansion, then we are now able to express $\bar{\boldsymbol{G}}_s$ formally as:

$$\bar{\boldsymbol{G}}_s = \frac{\boldsymbol{G}_s^{aggr}}{\boldsymbol{N}_s}.$$

Importantly, if the node s corresponds to the state reached from state $\hat{s}_\tau$ by performing action $\hat{a}_\tau$, then $\bar{G}(\hat{s}_\tau, \hat{a}_\tau) = \bar{\boldsymbol{G}}_s$ and $N(\hat{s}_\tau, \hat{a}_\tau) = \boldsymbol{N}_s$. The planning procedure described above ends when the maximum number of planning iterations is reached, or when a clear winner has been identified, i.e., if $\max_{a_t} P(a_t) - \frac{1}{\#A} > T_{dec}$ where $\#A$ is the number of possible actions, and T_{dec} is a (threshold) hyperparameter.

2.2.2 Derivation of the variational free energy

In this section, we provide a derivation for the variational free energy used by Fountas et al. (2020). This derivation was introduced in (Millidge, 2020), and can be adapted to derive the variational free energy of the models presented in Section 3. Recall, the goal of the variational free energy as classically presented is to make the approximate posterior $Q_\phi(s_t, a_t)$ as close as possible to the true posterior[3] $P(s_t, a_t | o_t, s_{t-1}, a_{t-1})$, i.e.,

$$Q^*_\phi(s_t, a_t) = \arg\min_{Q_\phi(s_t, a_t)} D_{\mathrm{KL}} \left[Q_\phi(s_t, a_t) || \, P(s_t, a_t | o_t, s_{t-1}, a_{t-1}) \right].$$

Using Bayes theorem, the linearity of expectation, and the fact that $Q_\phi(s_t, a_t)$ integrates to one:

$$\begin{aligned}
Q^*_\phi(s_t, a_t) &= \arg\min_{Q_\phi(s_t, a_t)} D_{\mathrm{KL}} \left[Q_\phi(s_t, a_t) || \, P(s_t, a_t | o_t, s_{t-1}, a_{t-1}) \right] \\
&= \arg\min_{Q_\phi(s_t, a_t)} D_{\mathrm{KL}} \left[Q_\phi(s_t, a_t) || \, P(s_t, a_t, o_t, s_{t-1}, a_{t-1}) \right] + \underbrace{\ln P(o_t, s_{t-1}, a_{t-1})}_{\text{Constant w.r.t } Q_\phi(s_t, a_t)} \\
&= \arg\min_{Q_\phi(s_t, a_t)} D_{\mathrm{KL}} \left[Q_\phi(s_t, a_t) || \, P(s_t, a_t, o_t, s_{t-1}, a_{t-1}) \right].
\end{aligned}$$

Using the d-separation criteria (Koller and Friedman, 2009), it ~~is straightforward to show~~ can be shown that:

$$P(s_t, a_t, o_t, s_{t-1}, a_{t-1}) = P_{\theta_o}(o_t | s_t) P(a_t) P_{\theta_s}(s_t | s_{t-1}, a_{t-1}) Q_\phi(s_{t-1}, a_{t-1}),$$

where $Q_\phi(s_{t-1}, a_{t-1})$ is the variational posterior obtained through the inference process at the previous time step. In the above equation, $Q_\phi(s_{t-1}, a_{t-1})$ was used to replace $P(s_{t-1}, a_{t-1})$, i.e., $Q_\phi(s_{t-1}, a_{t-1})$ was used as an empirical prior. Additionally, since $Q_\phi(s_{t-1}, a_{t-1})$ is a constant w.r.t $Q_\phi(s_t, a_t)$, the above minimization problem reduces to:

$$\begin{aligned}
Q^*_\phi(s_t, a_t) \quad &= \quad \arg\min_{Q_\phi(s_t, a_t)} \quad \underbrace{D_{\mathrm{KL}} \left[Q_\phi(s_t, a_t) || \, P_{\theta_o}(o_t | s_t) P(a_t) P_{\theta_s}(s_t | s_{t-1}, a_{t-1}) \right]}_{\text{variational free energy}} \qquad (5) \\
&= \arg\min_{Q_{\phi_a}(a_t | s_t) Q_{\phi_s}(s_t)} \quad \mathbb{E}_{Q_{\phi_s}(s_t)} \Big[D_{\mathrm{KL}} \left[Q_{\phi_a}(a_t | s_t) || \, P(a_t) \right] \Big] + D_{\mathrm{KL}} \left[Q_{\phi_s}(s_t) || \, P_{\theta_s}(s_t | s_{t-1}, a_{t-1}) \right] \\
&\qquad - \mathbb{E}_{Q_{\phi_s}(s_t)} \Big[\ln P_{\theta_o}(o_t | s_t) \Big].
\end{aligned}$$

By comparing the VFE in (5) and the EFE in (2) both from Fountas et al. (2020), one can see an inconsistency. Namely, the parameters are seen as latent variables in the EFE definition, c.f., θ in (2), but they are regarded as parameters of neural ~~netowrks~~ networks in the VFE, c.f., θ_s and θ_o in 5. This is important because θ cannot be both a parameter (i.e, real, vector, or matrix) and a random variable, and if θ is a random variable one must define its probability density, i.e., $P(\theta)$. Additionally, this inconsistency raises the question of whether the EFE is really the expectation of the VFE. To sum up, the DAI_{MC} agent is equipped with four deep neural networks modelling $Q_{\phi_a}(a_t | s_t)$, $Q_{\phi_s}(s_t)$, $P_{\theta_s}(s_t | s_{t-1}, a_{t-1})$, and $P_{\theta_o}(o_t | s_t)$. The weights of those networks are optimised using back-propagation to minimise the VFE given by (5). Note, (5) decomposes into two KL-divergence terms that can be computed analytically, and the expectations can be approximated using a Monte-Carlo estimate. Also, because $P_{\theta_o}(o_t | s_t)$ is modelled as a product of Bernoulli distributions, the logarithm of $P_{\theta_o}(o_t | s_t)$ reduces to the binary cross entropy.

[3] By posterior we mean a conditional distribution, where the given variables are those for which we observe a specific value. Note, the value of s_{t-1} is unknown but can be sampled from the posterior over s_{t-1} (from the previous action-perception cycle).

2.2.3 Independence assumptions and the expected free energy

The EFE as stated in Equation (2) needs to be re-arranged because it cannot be easily evaluated. ~~First, using~~ We therefore present the derivation proposed by Fountas et al. (2020). Then, we highlight two independence assumptions, i.e., $s_\tau \perp\!\!\!\perp \theta \mid \pi$ and $s_\tau \perp\!\!\!\perp \theta \mid \pi, o_\tau$, used without proof. Finally, we propose an alternative derivation that does not require those two assumptions and produces a simpler result. Using the product rule of probability, one can see that $Q(s_\tau, \theta|\pi) = Q(\theta|s_\tau, \pi)Q(s_\tau|\pi)$ and $\tilde{P}(o_\tau, s_\tau, \theta|\pi) = P(o_\tau|\pi)P(s_\tau|o_\tau, \pi)P(\theta|s_\tau, o_\tau, \pi)$. Using those two factorisations, the EFE given in (2), i.e.,

$$G_\tau(\pi) = \mathbb{E}_{\tilde{Q}}\Big[\ln Q(s_\tau, \theta|\pi) - \ln \tilde{P}(o_\tau, s_\tau, \theta|\pi)\Big],$$

where $\tilde{Q} = Q(o_\tau, s_\tau, \theta|\pi)$, and can be re-arranged as follows:

$$\begin{aligned} G_\tau(\pi) = &- \mathbb{E}_{\tilde{Q}}\Big[\ln \tilde{P}(o_\tau|\pi)\Big] \\ &+ \mathbb{E}_{\tilde{Q}}\Big[\ln Q(s_\tau|\pi) - \ln \tilde{P}(s_\tau|o_\tau, \pi)\Big] \\ &+ \mathbb{E}_{\tilde{Q}}\Big[\ln Q(\theta|s_\tau, \pi) - \ln \tilde{P}(\theta|s_\tau, o_\tau, \pi)\Big]. \end{aligned} \tag{6}$$

Note, the above derivation follows the work of Fountas et al. (2020).

Re-arranging the second term of Equation (6) according to Fountas et al. (2020)

~~In the supplementary material of (Fountas et al., 2020)~~First, the second term of Equation (6) is ~~re-aranged~~ re-arranged into entropy terms for which an analytical solution exists. In the supplementary material of (Fountas et al., 2020), the derivation goes as follows:

$$\begin{aligned} \mathbb{E}_{\tilde{Q}}\Big[\ln Q(s_\tau|\pi) - \ln \tilde{P}(s_\tau|o_\tau, \pi)\Big] &\triangleq \mathbb{E}_{\tilde{Q}}\Big[\ln Q(s_\tau|\pi) - \ln Q(s_\tau|o_\tau, \pi)\Big] \\ &= \mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)Q(o_\tau|s_\tau,\theta,\pi)}\Big[\ln Q(s_\tau|\pi) - \ln Q(s_\tau|o_\tau, \pi)\Big] \\ &= \mathbb{E}_{Q(\theta|\pi)}\Big[\mathbb{E}_{Q(s_\tau|\theta,\pi)}[\ln Q(s_\tau|\pi)] - \mathbb{E}_{Q(s_\tau|\theta,\pi)Q(o_\tau|s_\tau,\theta,\pi)}[\ln Q(s_\tau|o_\tau, \pi)]\Big] \\ &= \mathbb{E}_{Q(\theta|\pi)}\Big[\mathbb{E}_{Q(s_\tau|\theta,\pi)}[\ln Q(s_\tau|\pi)] - \mathbb{E}_{Q(o_\tau|\theta,\pi)Q(s_\tau,|o_\tau,\theta,\pi)}[\ln Q(s_\tau|o_\tau, \pi)]\Big], \end{aligned} \tag{7}$$

where in the first line a distribution was renamed, i.e., $\tilde{P}(s_\tau|o_\tau, \pi) \triangleq Q(s_\tau|o_\tau, \pi)$. The next step in the derivation (c.f. supplementals of Fountas et al. (2020)) re-arranges this final expression to the following:

$$\mathbb{E}_{\tilde{Q}}\Big[\ln Q(s_\tau|\pi) - \ln Q(s_\tau|o_\tau, \pi)\Big] = \mathbb{E}_{Q(\theta|\pi)}\Big[\mathbb{E}_{Q(o_\tau|\theta,\pi)}[H[Q(s_\tau|o_\tau, \pi)]] - H[Q(s_\tau|\pi)]\Big].$$

However, the above equation assumes that $s_\tau \perp\!\!\!\perp \theta \mid \pi$ and $s_\tau \perp\!\!\!\perp \theta \mid \pi, o_\tau$. In other words, some of the conditioning on θ has been dropped, i.e.,

$$\begin{aligned} &\mathbb{E}_{Q(\theta|\pi)}\Big[\mathbb{E}_{Q(s_\tau|\theta,\pi)}[\ln Q(s_\tau|\pi)] - \mathbb{E}_{Q(o_\tau|\theta,\pi)Q(s_\tau,|o_\tau,\theta,\pi)}[\ln Q(s_\tau|o_\tau, \pi)]\Big] \qquad \text{(last expression of derivation 7)} \\ &\neq \mathbb{E}_{Q(\theta|\pi)}\Big[\mathbb{E}_{Q(s_\tau|\pi)}[\ln Q(s_\tau|\pi)] - \mathbb{E}_{\underline{Q(o_\tau|\theta,\pi)Q(s_\tau,|o_\tau,\pi)}[\ln Q(s_\tau|o_\tau, \pi)]}{}_{Q(o_\tau|\theta,\pi)Q(s_\tau|o_\tau,\pi)}[\ln Q(s_\tau|o_\tau, \pi)]\Big] \\ &= \mathbb{E}_{Q(\theta|\pi)}\Big[\mathbb{E}_{Q(o_\tau|\theta,\pi)}\big[H[Q(s_\tau|o_\tau, \pi)]\big] - H[Q(s_\tau|\pi)]\Big]. \end{aligned}$$

Whether this conditioning can be dropped or not depends on the factorisation of the distribution. In other words, the two assumptions (i.e., $s_\tau \perp\!\!\!\perp \theta \mid \pi$ and $s_\tau \perp\!\!\!\perp \theta \mid \pi, o_\tau$) would have to be checked using the d-separation criterion. However, this is difficult to do, since as mentioned previously, the parameters θ are latent variables in (2) but are regarded as parameters in (5), which makes the graphical model unclear. Instead of attempting to prove that $s_\tau \perp\!\!\!\perp \theta \mid \pi$ and $s_\tau \perp\!\!\!\perp \theta \mid \pi, o_\tau$, we propose an alternative derivation that does not require such independence assumptions.

Alternative derivation of the second term of Equation (6)

Restarting from the second term of Equation (6), we can re-arrange as follows:

$$\begin{aligned}
\mathbb{E}_{\tilde{Q}}\Big[\ln Q(s_\tau|\pi) - \ln \tilde{P}(s_\tau|o_\tau,\pi)\Big] &\triangleq \mathbb{E}_{\tilde{Q}}\Big[\ln Q(s_\tau|\pi) - \ln Q(s_\tau|o_\tau,\pi)\Big]\\
&= \mathbb{E}_{Q(s_\tau|\pi)Q(\theta,o_\tau|s_\tau,\pi)}\Big[\ln Q(s_\tau|\pi)\Big] - \mathbb{E}_{Q(o_\tau|\pi)Q(s_\tau|o_\tau,\pi)Q(\theta|s_\tau,o_\tau,\pi)}\Big[\ln Q(s_\tau|o_\tau,\pi)\Big]\\
&= \mathbb{E}_{Q(s_\tau|\pi)}\Big[\ln Q(s_\tau|\pi)\Big] - \mathbb{E}_{Q(o_\tau|\pi)Q(s_\tau|o_\tau,\pi)}\Big[\ln Q(s_\tau|o_\tau,\pi)\Big]\\
&= \mathbb{E}_{Q(o_\tau|\pi)}\Big[H[Q(s_\tau|o_\tau,\pi)]\Big] - H[Q(s_\tau|\pi)],
\end{aligned}$$

where in the first line a distribution was renamed, i.e., $\tilde{P}(s_\tau|o_\tau,\pi) \triangleq Q(s_\tau|o_\tau,\pi)$, two different factorizations of $\tilde{Q}$ are used going from ~~first to~~ the first to the second line, the linearity of expectations was used between the first and second line, and the expectation w.r.t θ was dropped (between ~~line~~ lines two and three) because the expectation of a constant is the constant itself. Importantly, the above derivation does not make any assumption of independence, and leads to a simpler result. This alternative derivation is beneficial as it produces a stronger derivation because it makes less (unjustified) assumptions. The simpler result produced by this derivation also has a practical implication. Indeed, the expectation w.r.t. $Q(\theta|\pi)$ disapears and the expectation w.r.t. $Q(o_\tau|\theta,\pi)$ is now w.r.t. $Q(o_\tau|\pi)$. Those two changes sugguest that a different implementation of this term is required.

Re-arranging the third term of Equation (6) from Fountas et al. (2020)

~~We now,~~ For completeness, we now focus on the third term of (6), which can be re-arranged as follows:

$$\begin{aligned}
\mathbb{E}_{\tilde{Q}}\Big[\ln Q(\theta|s_\tau,\pi) - \ln \tilde{P}(\theta|s_\tau,o_\tau,\pi)\Big] &\triangleq \mathbb{E}_{\tilde{Q}}\Big[\ln Q(\theta|s_\tau,\pi) - \ln Q(\theta|s_\tau,o_\tau,\pi)\Big]\\
&= \mathbb{E}_{\tilde{Q}}\Big[\ln Q(o_\tau|s_\tau,\pi) - \ln Q(o_\tau|s_\tau,\theta,\pi)\Big],
\end{aligned}$$

where $\tilde{P}(\theta|s_\tau,o_\tau,\pi)$ was renamed as $Q(\theta|s_\tau,o_\tau,\pi)$, and Bayes theorem was used to get:

$$Q(\theta|s_\tau,\pi) = \frac{Q(\theta|s_\tau,o_\tau,\pi)Q(o_\tau|s_\tau,\pi)}{Q(o_\tau|s_\tau,\theta,\pi)} \Leftrightarrow \frac{Q(\theta|s_\tau,\pi)}{Q(\theta|s_\tau,o_\tau,\pi)} = \frac{Q(o_\tau|s_\tau,\pi)}{Q(o_\tau|s_\tau,\theta,\pi)}.$$

Finally, by recalling that $\tilde{Q} = Q(o_\tau,s_\tau,\theta|\pi) = Q(\theta|\pi)Q(s_\tau|\theta,\pi)Q(o_\tau|s_\tau,\theta,\pi)$, and using the linearity of expectation, we get:

$$\begin{aligned}
\mathbb{E}_{\tilde{Q}}\Big[\ln Q(\theta|s_\tau,\pi) - \ln Q(\theta|s_\tau,o_\tau,\pi)\Big] &= \mathbb{E}_{Q(o_\tau,s_\tau|\pi)}\Big[\ln Q(o_\tau|s_\tau,\pi)\Big] + \mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)}\Big[H[Q(o_\tau|s_\tau,\theta,\pi)]\Big]\\
&= \mathbb{E}_{Q(o_\tau|s_\tau,\pi)Q(s_\tau|\pi)}\Big[\ln Q(o_\tau|s_\tau,\pi)\Big] + \mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)}\Big[H[Q(o_\tau|s_\tau,\theta,\pi)]\Big]\\
&= -\mathbb{E}_{Q(s_\tau|\pi)}\Big[H\big[Q(o_\tau|s_\tau,\pi)\big]\Big] + \mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)}\Big[H\big[Q(o_\tau|s_\tau,\theta,\pi)\big]\Big],
\end{aligned}$$

where because $\ln Q(o_\tau|s_\tau,\pi)$ is a constant w.r.t θ, we have been able to use:

$$\mathbb{E}_{Q(o_\tau,\theta,s_\tau|\pi)}[\ln Q(o_\tau|s_\tau,\pi)] = \mathbb{E}_{Q(o_\tau,s_\tau|\pi)}[\ln Q(o_\tau|s_\tau,\pi)].$$

To sum up, this derivation provides an expression based on the following two entropy terms: $H[Q(o_\tau|s_\tau,\pi)]$ and $H[Q(o_\tau|s_\tau,\theta,\pi)]$, which Fountas et al. (2020) claim can be estimated (c.f. Apprndix B for more details). Note that our proposed alternative to the EFE (see below) uses this derivation for the third term of Equation (6).

The EFE from Fountas et al. (2020)

If one follows the derivation proposed by Fountas et al. (2020), then the EFE is given by:

$$\begin{aligned} G_\tau(\pi) = &- \mathbb{E}_{\tilde{Q}}\Big[\ln \tilde{P}(o_\tau|\pi)\Big] \\ &+ \mathbb{E}_{Q(\theta|\pi)}\Big[\mathbb{E}_{Q(o_\tau|\theta,\pi)}\big[H[Q(s_\tau|o_\tau,\pi)]\big] - H[Q(s_\tau|\pi)]\Big] \\ &+ \mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)}\Big[H[Q(o_\tau|s_\tau,\theta,\pi)]\Big] - \mathbb{E}_{Q(s_\tau|\pi)}\Big[H\big[Q(o_\tau|s_\tau,\pi)\big]\Big]. \end{aligned} \tag{8}$$

~~The way this expression is evaluated in practice is described~~ Note, in Section 2.2, we focused on presenting the approach of Fountas et al. (2020) — as presented in their paper — in the most consistent manner posible. More details about the implementation of Equation 8 is presented in Appendix B, along with some discrepancies between the paper and the code.

Our proposed alternative to the EFE

If one follows our alternative derivation, then the EFE is given by:

$$\begin{aligned} G_\tau(\pi) = &- \mathbb{E}_{\tilde{Q}}\Big[\ln \tilde{P}(o_\tau|\pi)\Big] \\ &+ \mathbb{E}_{Q(o_\tau|\pi)}\Big[H[Q(s_\tau|o_\tau,\pi)]\Big] - H[Q(s_\tau|\pi)], \\ &+ \mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)}\Big[H[Q(o_\tau|s_\tau,\theta,\pi)]\Big] - \mathbb{E}_{Q(s_\tau|\pi)}\Big[H\big[Q(o_\tau|s_\tau,\pi)\big]\Big]. \end{aligned}$$

~~Despite what we have seen so far, and the discrepancies described~~ As explained in Section 2.2.3, it is not clear how to prove the two independence assumptions (i.e., $s_\tau \perp\!\!\!\perp \theta \mid \pi$ and $s_\tau \perp\!\!\!\perp \theta \mid \pi, o_\tau$) used by Fountas et al. (2020) in their derivation of the expected free energy. Additionally, as presented in Appendix B, we were able to identify discrepancies between the paper and the code. Despite those two points, DAI_{MC} does solve the dSprites environment.

2.3 DAI_{VPG} agent (Millidge, 2020)

In this section, we explain and discuss the approach of Millidge (2020). The code is available at the following URL: `https://github.com/BerenMillidge/DeepActiveInference`. Note that, though the mathematics in the paper are based on the formalism of a partially observable Markov decision process (POMDP), the code does not implement an encoder/decoder architecture, which means that the code implements a fully observable decision process, i.e., MDP. Additionally, the DAI_{VPG} is composed of three neural networks illustrated in Figures 4 and 5. The first is the transition network that predicts the future observations based on the current observations and action, i.e., $\mathring{o}_{\tau+1} = \mathcal{T}_{\theta_o}(o_\tau, a_\tau)$. The second is the policy network that models the variational distribution over actions $Q_{\phi_a}(a_\tau|o_\tau)$. The third is the critic network that predicts the expected free energy of each action given the current observation. Moreover, Millidge (2020) defines the prior over actions as follows:

$$P(a_\tau|o_\tau) = \sigma[-\zeta G(o_\tau, a_\tau)],$$

where ζ is the precision of the prior over actions, $\sigma[\bullet]$ is a softmax function, and $G(o_\tau, a_\tau)$ is the expected free energy (EFE) of taking action a_τ when observing o_τ. In the paper, the mathematics are based on the POMDP formalism. Therefore, $G(o_\tau, a_\tau)$ is denoted $G(s_\tau, a_\tau)$, and is defined as follows:

$$G(s_\tau, a_\tau) = -r_\tau + \underbrace{D_{\mathrm{KL}}\left[\,Q(s_\tau)||\,Q(s_\tau|o_\tau)\right]}_{\text{intrinsic value}} + \hat{\mathcal{G}}_{\hat{\theta}_a}(a_{\tau+1}, s_{\tau+1}), \tag{9}$$

where r_τ is the reward gathered by the agent at time step τ, and $\hat{\mathcal{G}}_{\hat{\theta}_a}(a_{\tau+1}, s_{\tau+1})$ is the target network (i.e., a copy of the critic network whose weights are synchronised every K iterations of learning). Now, remember that in the implementation, there is no encoder $Q(s_\tau)$ and no decoder $P(o_\tau|s_\tau)$. In other words, there are no hidden states s_τ, raising some uncertainty about how the intrinsic value is computed. The code available on Github[4] at the following URL: `https://github.com/BerenMillidge/DeepActiveInference`, in the file `active_inference_with_Tmodel.jl` (see line 51) suggests that the following equation is used:

$$\text{intrinsic value} = \sum_i \Big[o_{\tau+1}[i] - \mathring{o}_{\tau+1}[i]\Big]^2, \tag{10}$$

where $o_{\tau+1}[i]$ is the i-th observation received at time step $\tau + 1$, and $\mathring{o}_{\tau+1}[i]$ is the (numerical) value of the i-th observation (at time step $\tau + 1$) predicted by the transition network. More formally, the above formulation for the intrisic value corresponds to the KL-divergence between two Gaussian distributions both having an identity covariance matrix, i.e.,

$$\text{intrinsic value} = D_{\text{KL}}\left[\, Q(o_{\tau+1}) || \, P(o_{\tau+1}|o_\tau, a_\tau)\right] = \sum_i \Big[o_{\tau+1}[i] - \mathring{o}_{\tau+1}[i]\Big]^2,$$

where $P(o_{\tau+1}|o_\tau, a_\tau) = \mathcal{N}(o_{\tau+1}; \mathring{o}_{\tau+1}, I)$ and $Q(o_{\tau+1})$ is a Gaussian distribution with mean vector $o_{\tau+1}$ and an identity covariance matrix. However, note that (9) is the definition of the expected free energy in the POMDP setting. As explained by Costa et al. (2020b), the expected free energy in the MDP setting is given by:

$$G(a_{t:T-1}, o_t) \approx \sum_{\tau=t+1}^{T} D_{\text{KL}}\left[\, P(o_\tau|a_{t:T-1}, o_t) || \, P(o_\tau)\right],$$

where $P(o_\tau)$ are the prior preferences of the agent (related to rewards in reinforcement learning), and $P(o_\tau|a_{t:T-1}, o_t)$ is the transition mapping. Importantly, this definition for the expected free energy does not ~~decomposes~~ decompose into extrinsic and intrinsic terms as in (9). Thus, (as it stands) the implementation of the DAI_{VPG} agent is a mixture between the POMDP and MDP setting, where the generative model corresponds to a MDP, and the expected free energy is adapted from the POMDP setting.

We conclude this section by dicussing the training procedure of the transition, policy and critic networks. As explained in the paper, the transition network is trained to minimise the variational free enery. Additionally, because of the Gaussian assumptions (with identity covariance matrices) mentioned above, the KL-divergence reduces to the mean square error (MSE). Thus, the transition network is updated to minimise the MSE between the observations made by the agent at time $\tau + 1$, and the observations ($\mathring{o}_{\tau+1}$) predicted by the transition network, i.e.,

$$\theta_o^* = \arg\min_{\theta_o} \text{MSE}\Big[o_{\tau+1}, \mathring{o}_{\tau+1}\Big],$$

where $\mathring{o}_{\tau+1} = \mathcal{T}_{\theta_o}(o_\tau, a_\tau)$. The policy network is trained to minimise the KL-divergence between the variational posterior over actions $Q_{\phi_a}(a_\tau|o_\tau)$ and the prior over actions $P(a_\tau|o_\tau)$, i.e.,

$$\phi_a^* = \arg\min_{\phi_a} D_{\text{KL}}\left[\, Q_{\phi_a}(a_\tau|o_\tau) || \, P(a_\tau|o_\tau)\right],$$

which minimises the variational free energy. Finally, the critic is trained by minimising the MSE between the target EFE as defined in (9) and the ouput of the critic $\mathcal{G}_{\theta_a}(o_\tau, \bullet)$:

$$\theta_a^* = \arg\min_{\theta_a} \text{MSE}\Big[G(o_\tau, \bullet), \mathcal{G}_{\theta_a}(o_\tau, \bullet)\Big].$$

DAI_{VPG} is able to solve the CartPole environment.

[4] We are referring to the version of the code that was available on github on the 6th of June 2022.

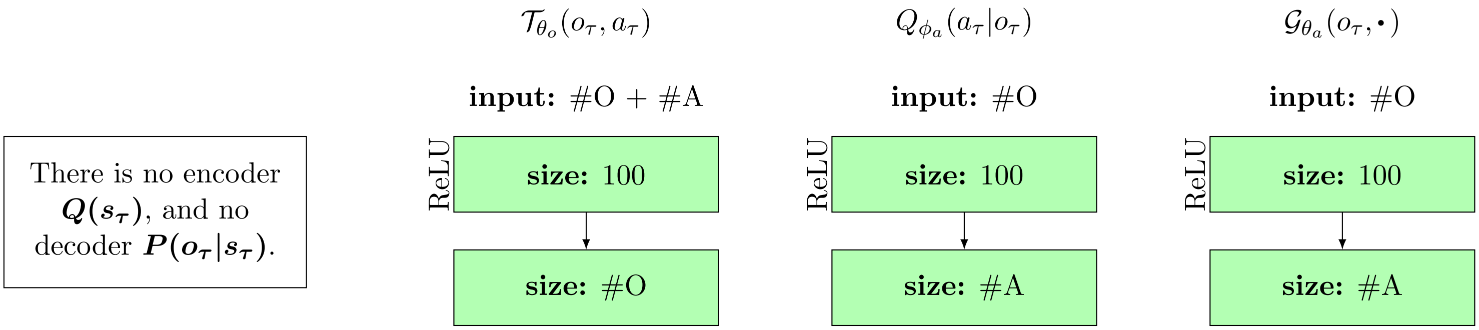

$\mathcal{T}_{\theta_o}(o_\tau, a_\tau)$
$Q_{\phi_a}(a_\tau|o_\tau)$
$\mathcal{G}_{\theta_a}(o_\tau, \bullet)$
There is no encoder $\boldsymbol{Q(s_\tau)}$, and no decoder $\boldsymbol{P(o_\tau|s_\tau)}$.
input: #O + #A
input: #O
input: #O
ReLU
size: 100
size: #O
ReLU
size: 100
size: #A
ReLU
size: 100
size: #A

Figure 4: Neural networks architecture of the DAI_{VPG} agent. Green blocks correspond to fully connected layers. The first neural network is the transition network that takes as input the observation and action at time step τ, and outputs the mean of a Gaussian distribution over observation at time step $\tau + 1$. The second neural network is the policy network that models the variational posterior over actions. The third neural network is the critic that takes as input an observation and outputs the expected free energy of each action.

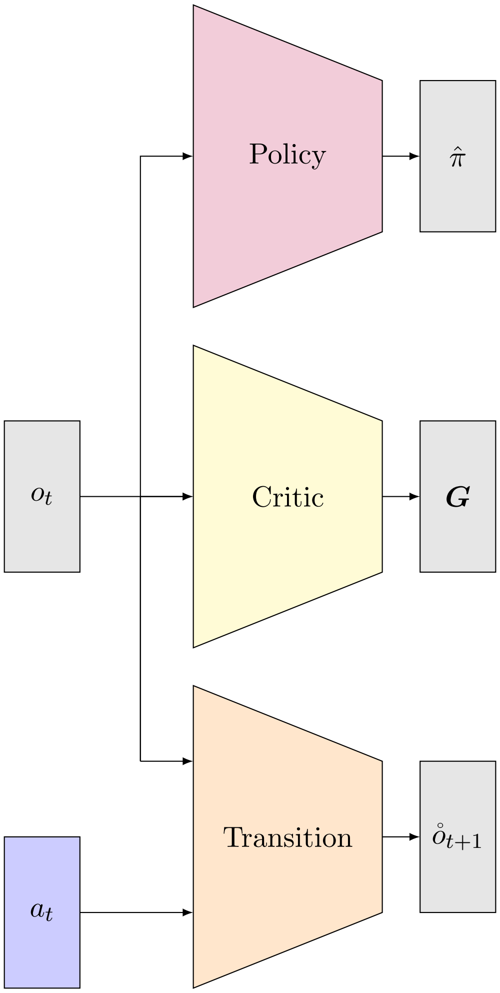


Figure 5: This figure illustrates the DAI_{VPG} agent. The only new part is the policy network, which takes as input the hidden state at time t and ouputs the ~~the~~ parameters $\hat{\pi}$ of the variational posterior over actions. Importantly, the DAI_{VPG} takes actions based on the EFE.

2.4 DAI_{RHI} agent (Rood et al., 2020)

In this section, we ~~quickly~~ explain and discuss the approach of Rood et al. (2020). Put simply, this paper proposes a variational auto-encoder (VAE), which is able to account for results that were observed in the context of the rubber-hand illusion (RHI) experiment. In the experiment in Rood et al. (2020), an agent (i.e., either a human or a computer) is able to move an arm in a 3D space. However, the agent does not observe the real position of the arm, instead, the agent sees an artificial hand placed in a different location. This can be implemented using virtual reality (for humans) or within a simulator (for computers). Since, Rood et al. (2020) restricted themself to the context of a VAE, this approach cannot be considered as a complete implementation of deep active inference. More precisely, the transition and critic (or policy) networks are missing.

2.5 DAI_{HR} agent (Sancaktar et al., 2020; Lanillos et al., 2020; Oliver et al., 2019)

In this section, we ~~quickly~~ explain and discuss the following approaches: Sancaktar et al. (2020), Lanillos et al. (2020), and Oliver et al. (2019). Briefly, those papers propose a free energy minimisation scheme based on a single decoder network, which is used to control Nao, TIAGo, and iCub robots, respectively. Since, Sancaktar et al. (2020); Lanillos et al. (2020); Oliver et al. (2019) restricted themself to the context of a single decoder, this approach can not be considered as a complete implementation of deep active inference. More precisely, the encoder, transition and critic (or policy) networks are missing.

2.6 DAI_{FA} agent (Ueltzhöffer, 2018)

In this section, we review the approach proposed by Ueltzhöffer (2018). The original code of this paper is available on GitHub at the following URL: `https://github.com/kaiu85/deepAI_paper`. This approach is composed of four deep neural networks. The encoder $\mathcal{E}_{\phi_s}$ models the approximate posterior over states $Q_{\phi_s}(s_t|s_{t-1}, o_t)$ as a Gaussian distribution, i.e., $Q_{\phi_s}(s_t|s_{t-1}, o_t) = \mathcal{N}(s_t; \mu, \sigma)$ where $\mu, \sigma = \mathcal{E}_{\phi_s}(s_{t-1}, o_t)$. The decoder $\mathcal{D}_{\theta_o}$ models the likelihood mapping $P_{\theta_o}(o_\tau|s_\tau)$ as a Gaussian distribution, i.e., $P_{\theta_o}(o_\tau|s_\tau) = \mathcal{N}(o_\tau; \mu_o, \sigma_o)$ where $\mu_o, \sigma_o = \mathcal{D}_{\theta_o}(s_\tau)$. The transition network $\mathcal{T}_{\theta_s}$ models the transition mapping $P_{\theta_s}(s_\tau|s_{\tau-1})$ as a Gaussian distribution, i.e., $P_{\theta_s}(s_\tau|s_{\tau-1}) = \mathcal{N}(s_\tau; \mathring{\mu}, \mathring{\sigma})$ where $\mathring{\mu}, \mathring{\sigma} = \mathcal{T}_{\theta_s}(s_{\tau-1})$. Note that, the transition network is only conditioned on the previous state~~, this~~. This is because the action is contained in the observations predicted by the decoder. ~~Thus, the latent space has to (implicitly) encode the action for the decoder to successfully predict them. Finally, the policy network $\mathcal{P}_{\theta_a}$ models the prior over actions $P_{\theta_a}(a_\tau|s_\tau)$ as a Gaussian distribution, i.e., $P_{\theta_a}(a_\tau|s_\tau) = \mathcal{N}(a_\tau; \mu_a, \sigma_a)$ where $\mu_a, \sigma_a = \mathcal{P}_{\theta_a}(s_\tau)$. Figure 6 illustrates the architectures of those deep neural networks. Importantly~~More precisely, the experiments were run in the MountainCar environment, which means that the agent is observing the x position of the car o^x_τ. Additionally, according to the idea of proprioception, the agent observes its own action, i.e., $o^a_{\tau-1} = a_{\tau-1}$ where $a_{\tau-1}$ is the action performed by the agent at time $\tau - 1$. In what follows, we let $o_\tau = (o^x_\tau, o^a_{\tau-1})$ be the concatenation of the x position of the car and the action taken by the agent. Importantly, because o_τ contains $o^a_{\tau-1}$, the latent space has to (implicitly) encode the action for the decoder to successfully predict the observations. Finally, the policy network $\mathcal{P}_{\theta_a}$ models the prior over actions $P_{\theta_a}(a_\tau|s_\tau)$ as a Gaussian distribution, i.e., $P_{\theta_a}(a_\tau|s_\tau) = \mathcal{N}(a_\tau; \mu_a, \sigma_a)$ where $\mu_a, \sigma_a = \mathcal{P}_{\theta_a}(s_\tau)$. Figure 6 illustrates the architectures of those deep neural networks. Then, Ueltzhöffer (2018) defines the free action objective as the cumulated variational free energy over time:

$$FA(o_{1:T}, \phi, \theta) = \sum_{\tau=1}^{T} \Bigg[\underbrace{\overbrace{-\mathbb{E}_{Q_{\phi_s}(s_\tau|s_{\tau-1}, o_\tau)}\Big[\ln P_{\theta_o}(o_\tau|s_\tau)\Big]}^{\text{accuracy}} + \overbrace{D_{\mathrm{KL}}\left[\, Q_{\phi_s}(s_\tau|s_{\tau-1}, o_\tau) || \, P_{\theta_s}(s_\tau|s_{\tau-1}) \right]}^{\text{complexity}}}_{\text{VFE}_\tau} \Bigg],$$

where $s_0 = (0, ..., 0)$ is a vector ~~full~~ of zeros representing the initial hidden state, T is the time horizon, $P_{\theta_o}(o_\tau|s_\tau)$, $Q_{\phi_s}(s_\tau|s_{\tau-1}, o_t)$, $P_{\theta_s}(s_\tau|s_{\tau-1})$ are modeled using Gaussian distributions whose parameters are predicted by the decoder, encoder and transition network, respectively. Figure 7 illustrates the computation of the free action objective, and the action-perception cycle of the agent. The first action-perception cycle is initiated when the intital hidden state s_0 is being fed into the policy network, which outputs the parameters of a Gaussian distribution over actions. Then, an action $\hat{a}_0$ is sampled from this Gaussian, and executed in the environment leading to a new observation o^x_1. Next, the action $\hat{a}_0$ is ~~concatenate~~ concatenated with o^x_1 to form o_1. The observation o_1 and the state s_0 are then fed into the encoder that outputs the parameters of a Gaussian distribution over $\hat{s}_1$. Lastly, a state is sampled from this Gaussian distribution and is used as input to the next action-perception cycle. This process continues until reaching the time horizon.

Within each action-perception cycle, the variational free energy of this time step is computed. To compute VFE_τ, the state s_τ is fed into both the tansition network and the encoder. Both networks output the parameters of a Gaussian distribution over $s_{\tau+1}$. A state is sampled from the distribution predicted by the encoder, and is used as input to the decoder that outputs the parameters of a Gaussian distribution over $o_{\tau+1}$. Finally, the parameters of the Gaussian distribution over $o_{\tau+1}$ is used to compute the accuracy term, and the parameters of the two Gaussian distributions over $s_{\tau+1}$ are used to compute the complexity term.

We now focus on the prior preferences of the agent. Usually, prior preferences are part of the expected free energy. However, Ueltzhöffer (2018) takes a different approach. Recall, the latent variable s_τ is modeled using a multivariate Gaussian. The DAI_{FA} agent reserves the first dimension of the latent space to the encoding of the prior preferences. Specifically, the transition network predicts the mean vector and the diagonal of the covariance matrix (i.e., another vector) of a multivariate Gaussian over latent states. The first element in the mean vector is clamped to the target x position, and the first element of the variance vector is set to a relatively small value. This effectively propels the agent towards the target location. Additionally, the encoder predicts another set of mean and variance vectors. The first element of the mean vector predicted by the encoder is clamped to the current x position observed by the agent, and the first element of the variance vector is set to a relatively small value. Note, clamping the value of the first element of the mean and variance vectors predicted by the transition is uncontentious, i.e., this is simply how the generative model is defined. However, clamping the value of the first element of the mean and variance vectors predicted by the encoder may be debated. Specifically, the encoder is supposed to predict the variational distribution, which is an approximation of the true posterior. However, clamping the value of the first element of the mean and variance vectors predicted by the encoder is likely to push the variational posterior further from the true posterior.

There are a number of important aspects of the DAI_{FA} agent. First, there is no expected free energy, instead the agent is trained to ~~minimised~~ minimise the cumulated variational free energy over time. Second, this approach unrolls the partially observable Markov decision process over time. In other words, the code builds a huge computational graph containing the encoder, decoder, transition and policy networks for each action-perception cycle. Therefore, the approach is computationally intensive and can become intractable for a large time horizon. Third, the DAI_{FA} requires the modeller to encode the prior preferences within the distributions predicted by the encoder and transition network. This can limit the applicability of the approach. Indeed, as previously explained, one can encode the prior preferences of the agent for the MountainCar problem within the first dimension of the latent space.

However, manually encoding the prior preferences in the latent space has two major drawbacks. First, the model needs to be modified from one environment to the next. This is because for each environment, the prior preferences of the agent will be different. Second, for some environments, it is unclear how the prior preferences may be defined. For example, when playing PacMan, the agent needs to eat all the dots, while simultaneously avoiding the ghosts. How can this be encoded in the model's latent space? This is particularly challenging because the only observation made by the agent is an image of the game, i.e., the agent does not directly have access to the positions of PacMan and the ghosts.

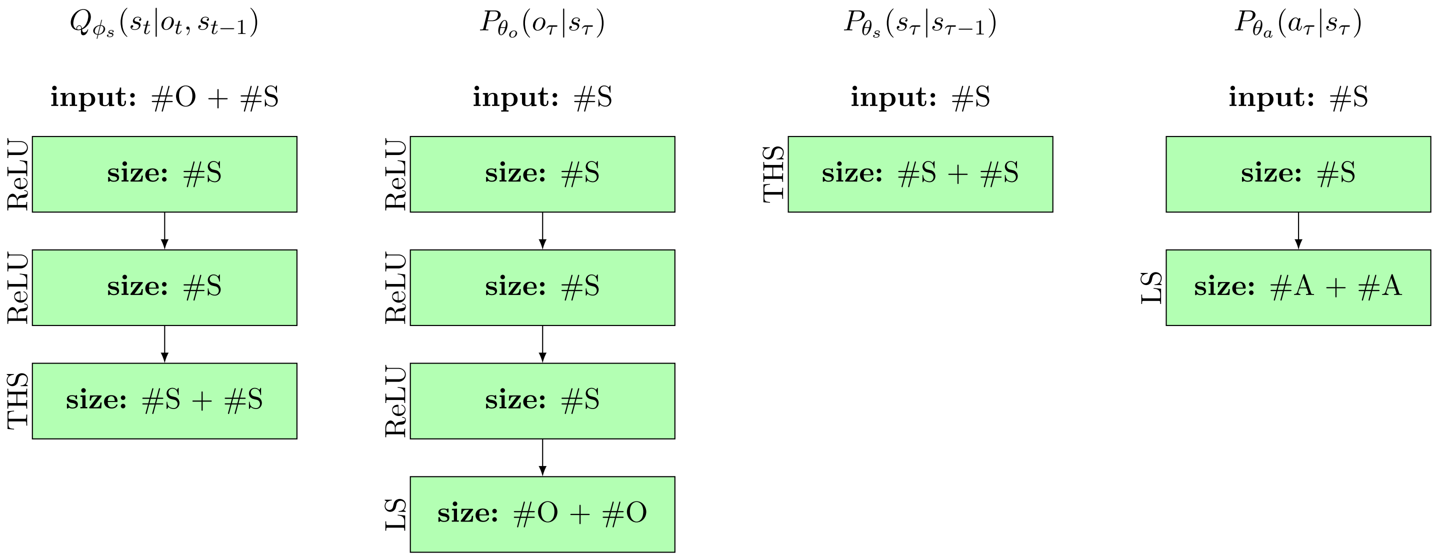


Figure 6: Neural networks architecture of the DAI_{FA} agent. Green blocks correspond to fully connected layers. The first neural network is the encoder that takes as input the state at time $t-1$ and the observation at time t, and outputs the parameters of a distribution over the state at time t. The second neural network is the decoder that takes as input the state at time τ, and outputs the parameters of a distribution over the observation at time τ. The third is the transition network that takes as input the state at time step $\tau - 1$, and outputs the parameters of a distribution over the state at time step τ. The fourth neural network is the policy network that models the prior over actions, i.e., the policy takes as input a state at time τ and outputs the parameters of a distribution over the actions at time step τ. Finally, THS stands for tangent hyperbolic and softplus, i.e., the tangent hyperbolic activation is over the first half of the neurons and the softplus activation function is over the second half, and LS stands for linear activation function and softplus, i.e., the linear activation is over the first half of the neurons and the softplus activation function is over the second half.

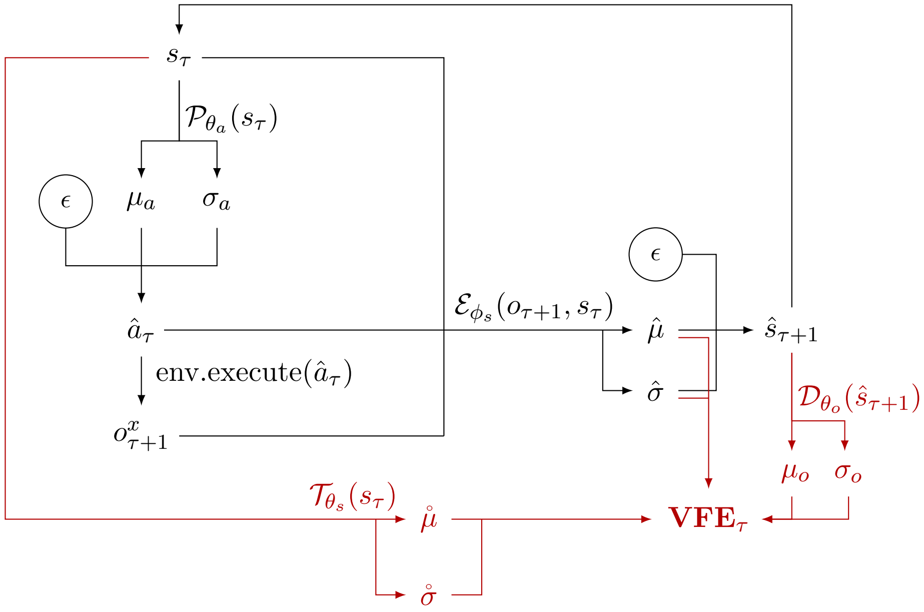


Figure 7: Action-perception cycles (in black) and estimation of the free action objective (in red). Note, $\mathring{\mu}$, $\mathring{\sigma}$, $\hat{\mu}$ and $\hat{\sigma}$ are used to compute the complexity terms of the variational free energy, while μ_o and σ_o are used to compute the accuracy term of the variational free energy.

2.7 DAI_{POMDP} agent (van der Himst and Lanillos, 2020)

In this section, we review the approach proposed by van der Himst and Lanillos (2020). The code is available at the following URL: `https://github.com/Grottoh/Deep-Active-Inference-for-Partially-Observable-MDPs`. The DAI_{POMDP} agent is composed of five deep neural networks.

The decoder $\mathcal{D}_{\theta_o}$ models $P_{\theta_o}(o_\tau|s_\tau)$ as a product of Bernoulli distributions, i.e., $P_{\theta_o}(o_\tau|s_\tau) = \mathcal{B}\text{ernoulli}(o_\tau; \hat{o}_\tau)$ where $\hat{o}_\tau = \mathcal{D}_{\theta_o}(s_\tau)$. The transition network $\mathcal{T}_{\theta_s}$ models $P_{\theta_s}(s_{\tau+1}|s_\tau, a_\tau)$ as a Gaussian distribution, i.e., $P_{\theta_s}(s_{\tau+1}|s_\tau, a_\tau) = \mathcal{N}(s_{\tau+1}|\mathring{\mu}, \mathring{\sigma})$ where $\mathring{\mu}, \ln \mathring{\sigma} = \mathcal{T}_{\theta_s}(s_\tau, a_\tau)$. The critic $\mathcal{G}_{\theta_a}$ outputs a vector containing the predicted expected free energy of each action, which is used to define the prior over action as $P_{\theta_a}(a_\tau|s_\tau) = \sigma[-\zeta \mathcal{G}_{\theta_a}(s_\tau, \bullet)]$, where $\sigma[\bullet]$ is a softmax function, ζ is the precision of the prior over actions, and $\mathcal{G}_{\theta_a}(s_\tau, \bullet)$ is the expected free energy of each action as predicted by the critic network when state s_τ is provided as input. The variational posterior over states $Q_{\phi_s}(s_t)$ is a Gaussian distribution modelled by the encoder $\mathcal{E}_{\phi_s}$, i.e., $Q_{\phi_s}(s_t) = \mathcal{N}(s_t; \mu, \sigma)$ where $\mu, \ln \sigma = \mathcal{E}_{\phi_s}(o_t)$. The variational posterior over actions $Q_{\phi_a}(a_t|s_t)$ is a categorical distribution modelled by the policy network $\mathcal{P}_{\phi_a}$, i.e., $Q_{\phi_a}(a_t|s_t) = \text{Cat}(a_t; \hat{\pi})$ where $\hat{\pi} = \mathcal{P}_{\phi_a}(s_t)$. Then, the agent is supposed to minimise the variational free energy defined as follows:

$$\begin{aligned}
Q^*_\phi(s_t, a_t) &= \underset{Q_\phi(s_t,a_t)}{\arg\min}\, D_{\text{KL}}\left[\, Q_{\phi_a}(a_t|s_t) Q_{\phi_s}(s_t) ||\, P_{\theta_o}(o_t|s_t) P_{\theta_s}(s_t|s_{t-1}, a_{t-1}) P_{\theta_a}(a_t|s_t)\right] \\
&= \underset{Q_\phi(s_t,a_t)}{\arg\min}\, D_{\text{KL}}\left[\, Q_{\phi_s}(s_t) ||\, P_{\theta_s}(s_t|s_{t-1}, a_{t-1})\right] + D_{\text{KL}}\left[\, Q_{\phi_a}(a_t|s_t) ||\, P_{\theta_a}(a_t|s_t)\right] - \mathbb{E}_{Q_{\phi_s}(s_t)}[\ln P_{\theta_o}(o_t|s_t)].
\end{aligned}$$

However, as explained in the paper, the KL-divergence (over states) is replaced by the mean square error (MSE) as follows:

$$Q^*_\phi(s_t, a_t) = \underset{Q_\phi(s_t,a_t)}{\arg\min}\, \text{MSE}(\mu, \mathring{\mu}) + D_{\text{KL}}\left[\, Q_{\phi_a}(a_t|s_t) ||\, P_{\theta_a}(a_t|s_t)\right] - \mathbb{E}_{Q_{\phi_s}(s_t)}[\ln P_{\theta_o}(o_t|s_t)],$$

where μ and $\mathring{\mu}$ are the mean vectors predicted by the encoder and the transition network, respectively. The paper justifies this substitution by saying that the maximum a posteriori (MAP) estimate is used to compute the state prediction error, instead of using the KL-divergence over the densities. However, the state prediction error and the KL-divergence over states are two different quantities, which are only equal when the two densities over states are Gaussian distributions with identity covariance matrix. However, the distribution predicted by the encoder network does not have an identity covariance matrix.

Put simply, in this context, the MSE and the KL-divergence between the densities over state are not necessarily equivalent. As a result, the DAI_{POMDP} agent may not always follow the free energy principle.

2.8 DAI_{SSM} agent (van der Himst and Lanillos, 2020)

The deep active inference agent proposed by (Çatal et al., 2020) is based on a state space model, and is therefore called DAI_{SSM}. The code of this approach was not available online, but we were able to retrieve it from the authors. Importantly, DAI_{SSM} is an offline approach meaning that the model is trained first on a fixed dataset gathered either by taking random actions in the environment or by manually controlling the robot. Then, when the model is trained, the expected free energy of different sequences of actions can be estimated using imaginary rollouts, and the first action of the policy with the lowest expected free energy is executed in the environment.

2.9 Unavailable code

The last paper (Schneider et al., 2022) was not reviewed because we have been unable to obtain the source code on the internet.

2.10 Representational similarity with centered kernel alignment

The goal of representational similarity metrics is, as its name indicates, to measure the similarity between two representations. In the context of deep learning, these representations correspond to $\mathbb{R}^{n \times p}$ matrices of activations, where n is the number of data examples and p the number of neurons in a layer. In this paper, we aim to use such metrics to compare the representations learned by the deep learning models described in Section 3 and the representations learned by a DQN.

For our analysis, we will use Centred Kernel Alignment (CKA) (Cortes et al., 2012; Cristianini et al., 2002), a normalised version of the Hillbert-Schmidt Independence Criterion (HSIC) (Gretton et al., 2005), which measures the alignment between the $n \times n$ kernel matrices of two representations. Kornblith et al. (2019) have shown that for deep learning applications, linear kernels with centred layer activations worked well. We thus focus on the linear CKA, also known as RV-coefficient (Robert and Escoufier, 1976). Moreover, it has been shown to provide results similar to other representational similarity metrics, while being faster to compute (Bonheme and Grzes, 2022).

For conciseness, we will refer to linear CKA as CKA in the rest of this paper. We now define CKA more formally. Given the centered layer activations $x \in \mathbb{R}^{n \times m}$ and $y \in \mathbb{R}^{n \times p}$ taken over n data examples, CKA is defined as:

$$CKA(x, y) = \frac{\|y^T x\|_F^2}{\|x^T x\|_F \|y^T y\|_F},$$

where $\|\cdot\|_F$ is the Frobenius norm, which is defined as:

$$\|a\|_F = \sqrt{\mathrm{tr}(aa^T)} = \sqrt{\sum_{i=1}^{k} \sum_{j=1}^{l} |a_{ij}|^2},$$

where $a \in \mathbb{R}^{k \times l}$ is an arbitrary $k \times l$ matrix, and a^T is the transpose of a.

Limitations of CKA While CKA leads to accurate results in practice, it can be overly sensitive to differences in neural architectures ~~Maheswaranathan et al. (2019)~~(Maheswaranathan et al., 2019), and can thus underestimate the similarity between activations coming from layers of different type (e.g., convolutional and deconvolutional). Thus, we will only discuss the variation of similarity when analysing such cases. For example, we will not compare $CKA(a, b)$ and $CKA(a, c)$ if a and b are convolutional layers but c is linear. We will, however, compare $CKA(a, c)$ and $CKA(b, c)$.

3. Incrementally building a deep active inference agent

All of the deep active inference models we have presented make important contributions, illustrating a range of possible implementations. However, we do not feel that any of these approaches is a complete and definitive realisation of deep active inference. We have highlighted limitations of these published approaches throughout our presentation. Accordingly, in the remainder of this paper, we step back to first principles and "~~build-up~~build up" an agent component-by-component to determine which parts of a "natural" deep active inference framework underlie its capacity to solve or fail to solve inference problems. Additionally, throughout this component-by-component investigation, we compare the different variants of deep active inference that result with a standard (~~well attested~~well-attested) approach: a deep Q-network (Mnih et al., 2013). Thus, in this section, we progresively build a deep active inference agent. Section 3.1 presents the dSprites environment in which all our simulations will be run. This environment was picked to test whether an active inference agent is able to solve the dSprites environment as previously claimed in the literature (Fountas et al., 2020). Section 3.2 describes how the agents introduced later in this paper interact with the environment. Then, Section 3.3 introduces a variational auto-encoder (VAE) agent, Section 3.4 discusses a deep hidden Markov model (HMM) agent, Section 3.5 presents a deep critical HMM (CHMM) agent, and finally, Section 3.6 introduces a complete deep active inference agent. Note, the notation used throughout this section are summarised in Appendix A.

3.1 dSprites environment

The dSprites environment is based on the dSprites dataset (Matthey et al., 2017), initially designed for analysing the latent representation learned by variational auto-encoders (Doersch, 2016). The dSprites dataset is composed of images of squares, ellipses and hearts. Each image contains one shape (square, ellipse or heart) with its own scale, orientation, and (X, Y) position. In the dSprites environment, the agent is able to move those shapes around by performing four actions (i.e., UP, DOWN, LEFT, RIGHT). To make the task tractable, the action selected by the agent is executed eight times in the environment before the beginning of the next action-perception cycle, i.e., the X or Y position is increased or decreased by eight between time step t and $t+1$. The goal of the agent is to move all squares towards the bottom-left corner of the image and all ellipses and hearts towards the bottom-right corner of the image, c.f. Figure 8.

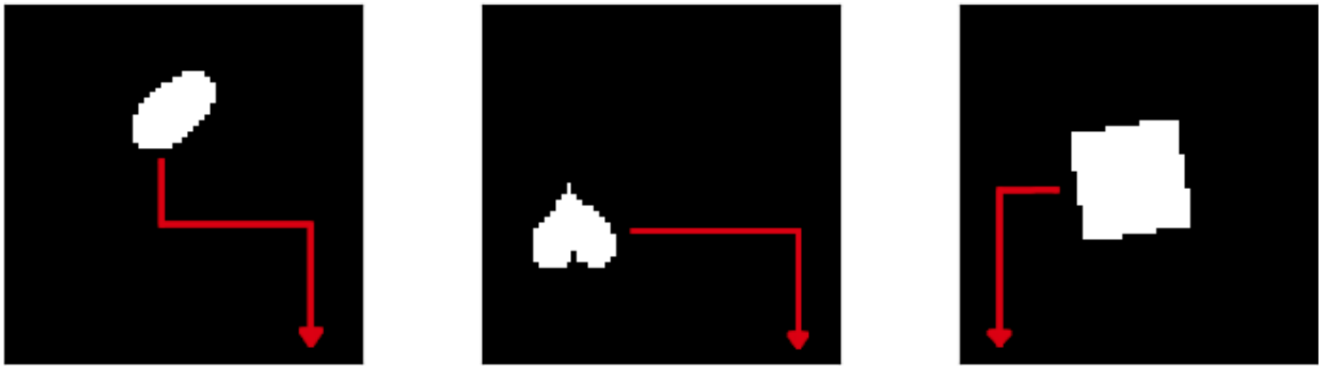


Figure 8: This figure illustrates the dSprites environment, in which the agent must move all squares towards the bottom-left corner of the image and all ellipses and hearts towards the bottom-right corner of the image. The red arrows show the behaviour expected from the agent.

3.2 Agent-environment interaction

In this section, we present how all the agents introduced in the next sections interact with the environment. Each agent was trained for $N = 500K$ iterations. At the begining of a trial, the environment is reset to a random state and the agent receives an observation[5] o_t. Using o_t, the agent selects an action a_t, which is then executed in the environment. This leads the agent to receive a new obervation o_{t+1}, a reward r_{t+1} and a boolean *done* describing whether the trial is over or not. Then, the new experience (o_t, a_t, o_{t+1}, r_{t+1}, *done*) is added to the replay buffer, from which a batch is sampled to train the various neural networks of the agent. Finally, if the trial has ended, then the environment is reset to a random state leading to a new observation o_t, otherwise o_{t+1} becomes the new o_t closing the action-perception cycle. Algorithm 1 summarises the agent-environment interaction.

[5] Each observation contains a sequence of three images, i.e., the image corresponding to the current state of the environment, and the two images gathered during the previous two time steps.

Algorithm 1: The interaction between the agent and the environment.

```
Input: $env$ the environment,
       $agent$ the agent,
       $buffer$ the replay buffer,
       $N$ the number of training iterations.

$o_t$ = env.reset()                                           // Get the initial observation from environment
repeat $N$ times
    $a_t \leftarrow$ select_action($o_t$)                                 // Select an action
    $o_{t+1}, r_{t+1}, done \leftarrow$ env.execute($a_t$)                // Execute the action in the environment
    buffer.push_new_experience($o_t$, $a_t$, $o_{t+1}$, $r_{t+1}$, $done$)  // Add the experience to the replay buffer
    agent.learn(buffer)                                       // Perform one iteration of training
    if $done == True$ then
        $o_t \leftarrow$ env.reset()                                      // Reset the environment when a trial ends
    else
        $o_t \leftarrow o_{t+1}$
    end
end
```

3.3 Variational auto-encoder

In this section, we present our first agent based on a variational auto-encoder. The agent is composed of two deep neural networks, i.e., an encoder and a decoder. The encoder $\mathcal{E}_{\phi_s}$ takes as input an image o_t and outputs the parameters of the variational posterior $Q_{\phi_s}(s_t) = \mathcal{N}(s_t; \mu, \sigma)$, where μ is the mean vector of the Gaussian distribution, and σ are the diagonal elements of the covariance matrix. The decoder $\mathcal{D}_{\theta_o}$ models the likelihood mapping $P_{\theta_o}(o_t|s_t)$, which attributes a probability to each image o_t given a state s_t, and is defined as:

$$P_{\theta_o}(o_t|s_t) = \mathcal{B}\text{ernoulli}(o_t; \hat{o}_t),$$

where $\hat{o}_t = \mathcal{D}_{\theta_o}(s_t)$ are the values predicted by the decoder, and $\mathcal{B}\text{ernoulli}(o_t; \hat{o}_t)$ is a product of Bernoulli distributions defined as:

$$\mathcal{B}\text{ernoulli}(o_t; \hat{o}_t) = \prod_{x,y} \text{Bernoulli}(o_t[x, y]; \hat{o}_t[x, y]),$$

where Bernoulli$(\bullet; \bullet)$ is a Bernoulli distribution over the possible values of the pixel $o_t[x, y]$, parameterized by the parameter $\hat{o}_t[x, y]$, which is predicted by the decoder network. The goal of the agent is to minimise the variational free energy (VFE):

$$\boldsymbol{F} = D_{\text{KL}}\left[\, Q_{\phi_s}(s_t) || \, P_{\theta_o}(o_t, s_t)\right] = D_{\text{KL}}\left[\, Q_{\phi_s}(s_t) || \, P_{\theta_o}(o_t|s_t)P(s_t)\right],$$

where $P(s_t) = \mathcal{N}(s_t; 0, I)$ is an isotropic (multivariate) Gaussian with variance one. The VFE can be re-arranged as follows:

$$\boldsymbol{F} = D_{\text{KL}}\left[\, Q_{\phi_s}(s_t) || \, P(s_t)\right] - \mathbb{E}_{Q_{\phi_s}(s_t)}[\ln P_{\theta_o}(o_t|s_t)],$$

where the KL-divergence between two Gaussian distributions can be computed using an analytical solution, and the expectation of the logarithm of $P_{\theta_o}(o_t|s_t)$ is approximated by a Monte-Carlo estimate using a single sample $\hat{s}_t \sim Q_{\phi_s}(s_t)$. The sample $\hat{s}_t$ is obtained using the reparameterisation trick as follows: $\hat{s}_t = \mu + \sigma \odot \hat{\epsilon}$, where $\odot$ is an element-wise product between two vectors, and $\hat{\epsilon} \sim \mathcal{N}(\epsilon; 0, I)$.

To sum up, this agent takes random actions, and stores its experiences in a ~~queue~~ replay buffer (c.f. Section 3.2). Then, batches of experiences (o_t, a_t, o_{t+1}, r_{t+1}, $done$) are sampled from the ~~queue~~replay buffer. The observations at time step t are then fed into the encoder, which outputs the mean and log variance of a Gaussian distribution $Q_{\phi_s}(s_t) = \mathcal{N}(s_t; \mu, \sigma)$. A latent state is sampled from $Q_{\phi_s}(s_t)$ using the re-parameterisation trick, and is then provided as input to the decoder which outputs the parameters of Bernoulli distributions $\hat{o}_t$. The KL-divergence between $Q_{\phi_s}(s_t)$ and $P(s_t)$ is computed analytically, and the logarithm of $P_{\theta_o}(o_t|s_t)$ reduces to the binary cross entropy (BCE) because $P_{\theta_o}(o_t|s_t)$ is a product of Bernoulli distributions. Next, the VFE is obtained by subtracting the BCE from the KL-divergence, and back-propagation is used to update the weights of the encoder and decoder networks. Figure 9 illustrates the VAE agent presented in this section. Note, this agent takes random actions.

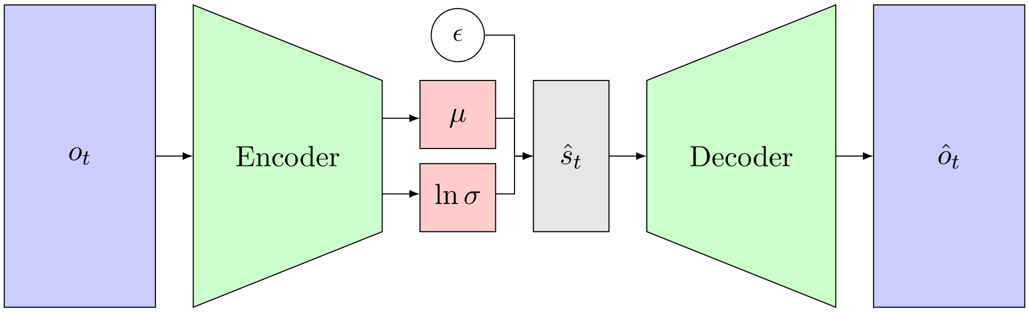


Figure 9: This figure illustrates the VAE agent. From left to right, we have the input image o_t, the encoder network, the layer of mean μ and log variance $\ln \sigma$, the epsilon random variable used for the reparameterisation trick, the latent state $\hat{s}_t$, the decoder network, and finally, the reconstructed image $\hat{o}_t$. Note, there are no actions in this agent's generative model. Therefore, the VAE agent takes random actions.

3.4 Deep hidden Markov model

In this section, we present our second agent based on a hidden Markov model. Similarly to the VAE agent, the HMM agent is composed of an encoder network modelling $Q_{\phi_s}(s_\tau)$, and a decoder network modelling $P_{\theta_o}(o_\tau|s_\tau)$. However, the prior over the hidden states at time step $t+1$ depends on the hidden states and action at time step t. This prior is modelled by the transition network $\mathcal{T}_{\theta_s}$ that predicts the parameters of the Gaussian distribution $P_{\theta_s}(s_{t+1}|s_t, a_t) = \mathcal{N}(s_{t+1}; \mathring{\mu}, \mathring{\sigma})$, where $\mathring{\mu}$ is the mean of the Gaussian distribution, and $\mathring{\sigma}$ are the diagonal elements of the covariance matrix. Recall, that the goal of the inference process is to fit the approximate posterior $Q_{\phi_s}(s_t)$ to the true posterior $P(s_t|o_t, s_{t-1}, a_{t-1})$. Formally, this optimisation can be written as the minimization of the Kullback-Leibler divergence between the approximate and the true posterior, i.e.,

$$Q^*(s_t) = \underset{Q_{\phi_s}(s_t)}{\arg\min}\, D_{\mathrm{KL}}\left[\, Q_{\phi_s}(s_t) ||\, P(s_t|o_t, s_{t-1}, a_{t-1})\right].$$

Using a derivation almost identical to the one presented in Section 2.2.2, the VFE can be proven to be:

$$\begin{aligned} Q^*_{\phi_s}(s_t) &= \underset{Q_{\phi_s}(s_t)}{\arg\min} \quad \underbrace{D_{\mathrm{KL}}\left[\, Q_{\phi_s}(s_t) ||\, P_{\theta_o}(o_t|s_t) P_{\theta_s}(s_t|s_{t-1}, a_{t-1})\right]}_{\text{variational free energy}} \\ &= \underset{Q_{\phi_s}(s_t)}{\arg\min} \quad D_{\mathrm{KL}}\left[\, Q_{\phi_s}(s_t) ||\, P_{\theta_s}(s_t|s_{t-1}, a_{t-1})\right] - \mathbb{E}_{Q_{\phi_s}(s_t)}[P_{\theta_o}(o_t|s_t)]. \end{aligned} \tag{11}$$

The VFE of Equation 11 can be computed in a similar way to the VAE agent. Put simply, this agent takes random actions, and stores its experiences in a ~~queue~~ replay buffer (c.f. Section 3.2). Then, batches of experiences (o_{t-1}, a_{t-1}, o_t, r_t, *done*) are sampled from the ~~queue~~replay buffer. The observations at time step $t-1$ are feed into the encoder, which outputs the mean and log variance of a Gaussian distribution $Q_{\phi_s}(s_{t-1}) = \mathcal{N}(s_{t-1}; \mu, \sigma)$. A latent state is sampled from $Q_{\phi_s}(s_{t-1})$ using the re-parameterisation trick, and is then provided as input to the transition network along with action a_{t-1}. The transition network outputs the parameters of the Gaussian distributions $P_{\theta_s}(s_t|s_{t-1}, a_{t-1}) = \mathcal{N}(s_t; \mathring{\mu}, \mathring{\sigma})$. Additionally, the observations at time step t can be fed into the encoder to get the parameters of $Q_{\phi_s}(s_t) = \mathcal{N}(s_t; \hat{\mu}, \hat{\sigma})$. Sampling from $Q_{\phi_s}(s_t)$ using the reparameterisation trick gives a state $\hat{s}_t$ that when given as input to the decoder produces the parameters of a product of Bernoulli distributions $\hat{o}_{t+1}$. The KL-divergence between $Q_{\phi_s}(s_t)$ and $P_{\theta_s}(s_t|s_{t-1}, a_{t-1})$ is computed analytically, and the logarithm of $P_{\theta_o}(o_t|s_t)$ reduces to the ~~BCE~~ binary cross entropy (BCE) because $P_{\theta_o}(o_t|s_t)$ is a product of Bernoulli distributions. Next, the VFE is obtained by subtracting the BCE from the KL-divergence, and back-propagation is used to update the weights of the encoder, decoder and transition networks. Figure 10 illustrates the HMM agent.

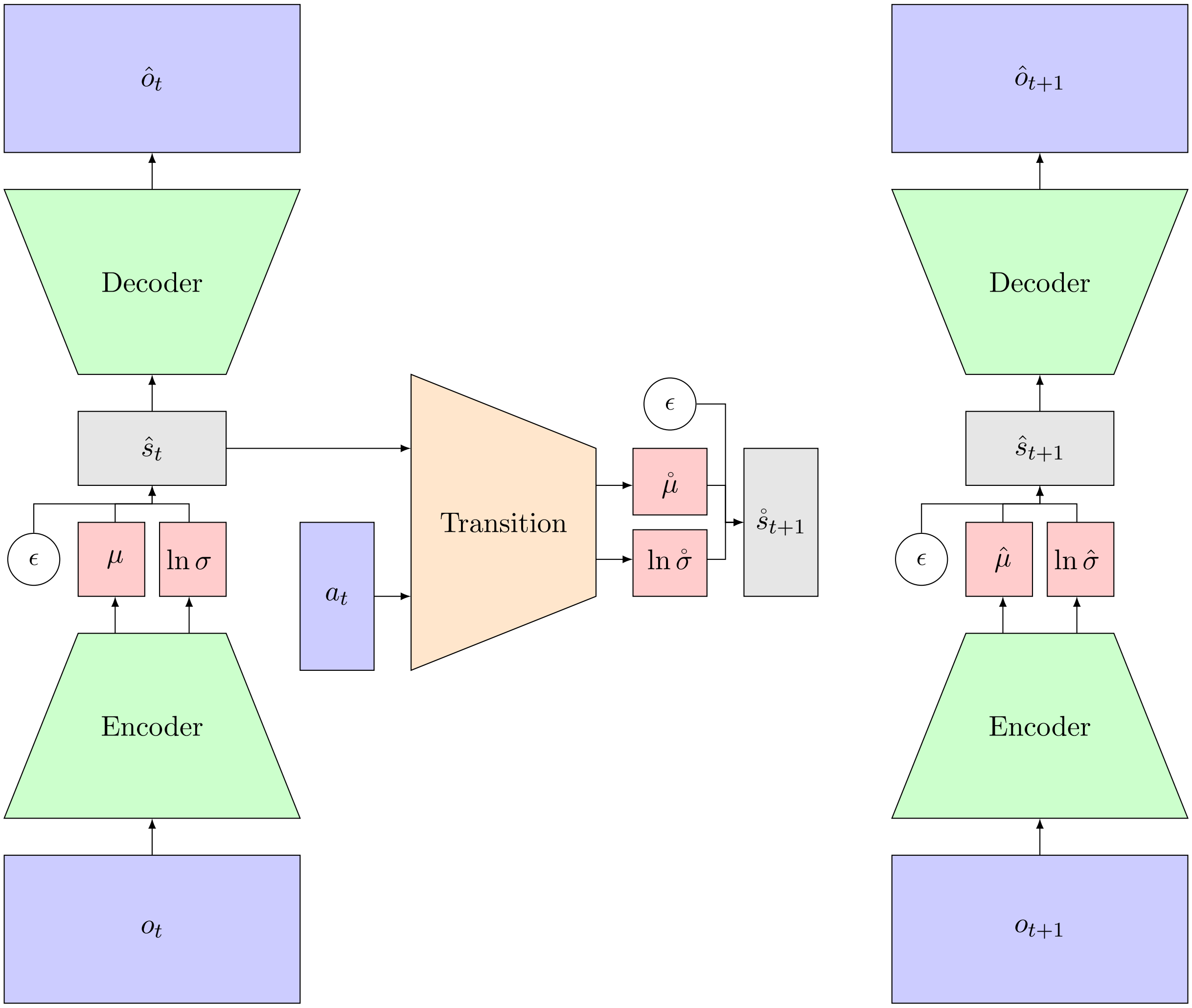


Figure 10: This figure illustrates the HMM agent. On the left and right, one can see two auto-encoders, i.e., one at time step t and one at time step $t+1$. In the middle, the transition network takes as input the state and action at time t, i.e., $(\hat{s}_t, a_t)$, and outputs the mean $\mathring{\mu}$ and log variance $\ln\mathring{\sigma}$ of a Gaussian distribution. By sampling the latent variable ϵ, and using the reparameterisation trick, we get the latent state outputed by the transition network: $\mathring{s}_{t+1} = \mathring{\mu} + \mathring{\sigma} \odot \hat{\epsilon}$ where $\hat{\epsilon}$ is sampled from a Gaussian distribution with mean zero and variance one. Importantly, the model seems to be composed of two disconnected parts, however, the variational free energy will have a complexity term between the Gaussian distributions outputed by the transition network and encoder at time $t+1$. Note, this agent takes random actions.

3.5 Deep critical HMM

In this section, we present our third agent that incorporates a critic network to the deep HMM presented in the previous section. The resulting model is called a deep CHMM and is illustrated in Figure 11. The CHMM is equipped with an encoder $\mathcal{E}_{\phi_s}$ modelling $Q_{\phi_s}(s_\tau)$, a decoder $\mathcal{D}_{\theta_o}$ modelling $P_{\theta_o}(o_\tau|s_\tau)$, a transition network $\mathcal{T}_{\theta_s}$ modelling $P_{\theta_s}(s_t|s_{t-1}, a_{t-1})$, and a critic network $\mathcal{G}_{\theta_a}$ that predicts the expected free energy (see below) of each action. The critic is then used to define the prior over actions as: $P_{\theta_a}(a_t|s_t) = \sigma[-\zeta\mathcal{G}_{\theta_a}(s_t, \bullet)]$, where ζ is the precision of the prior over actions, and $\mathcal{G}_{\theta_a}(s_t, \bullet)$ is the EFE of taking each action in state s_t as predicted by the critic. The encoder, decoder and transition networks are all trained in the same way as before to minimise the VFE. The critic however is trained to minimise the smooth L1 norm between its output $\mathcal{G}_{\theta_a}(s_t, \bullet)$ and the target G-values $y(\bullet)$, i.e., the critic minimises $\text{SL1}[\mathcal{G}_{\theta_a}(s_t, \bullet), y(\bullet)]$. Note, the SL1 was picked because it is less sensitive to outliers than the

MSE, and is defined as:

$$SL1[x,y] = \begin{cases} 0.5 \times \frac{(x-y)^2}{\beta} & \text{if } |x-y| < \beta \\ |x-y| - 0.5 \times \beta & \text{otherwise} \end{cases}.$$

Addtionally, the target G-values are defined as:

$$y(a_t) = G_{t+1}(a_t) + \gamma \mathbb{E}_{Q_{\phi_s}(s_{t+1})}\Big[\max_{a_{t+1} \in \mathcal{A}} \hat{\mathcal{G}}_{\hat{\theta}_a}(s_{t+1}, a_{t+1})\Big],$$

where $Q_{\phi_s}(s_{t+1})$ can be computed by feeding the image o_{t+1} sampled from the ~~queue~~ replay buffer as input to the encoder, $G_{t+1}(a_t)$ is the expected free energy at time $t+1$ after taking action a_t (see below), and γ is a discount factor. Note, the above Equation is an application on Bellman's equation (Bellman, 1952) to the expected free energy. Also, as for the DQN agent, we improved the training stability by implementing a target network $\hat{\mathcal{G}}_{\hat{\theta}_a}$, which is structurally identical to the critic and whose weights are synchronised with the weights of the critic every K (learning) iterations. The last question to answer before focusing on the subject of the EFE is: how does the CHMM select the action to be performed in the environment?

There are at least four possibilities: (i) select a random action, (ii) select the action that maximises EFE according to the critic, i.e., $a_t^* = \arg\max_{a_t} \mathcal{G}_{\theta_a}(s_t, a_t)$, (iii) sample an action from a softmax function of the output of the critic, i.e., $a_t^* \sim \sigma[\mathcal{G}_{\theta_a}(s_t, \bullet)]$ where $\sigma[\bullet]$ is a softmax function, and (iv) use the $\mathring{\epsilon}$-greedy algorithm with exponential decay, i.e., select a random action with probabilty $\mathring{\epsilon}$ or select the best action with probability $1 - \mathring{\epsilon}$ where $\mathring{\epsilon}$ starts with a high value and decays exponentially fast.

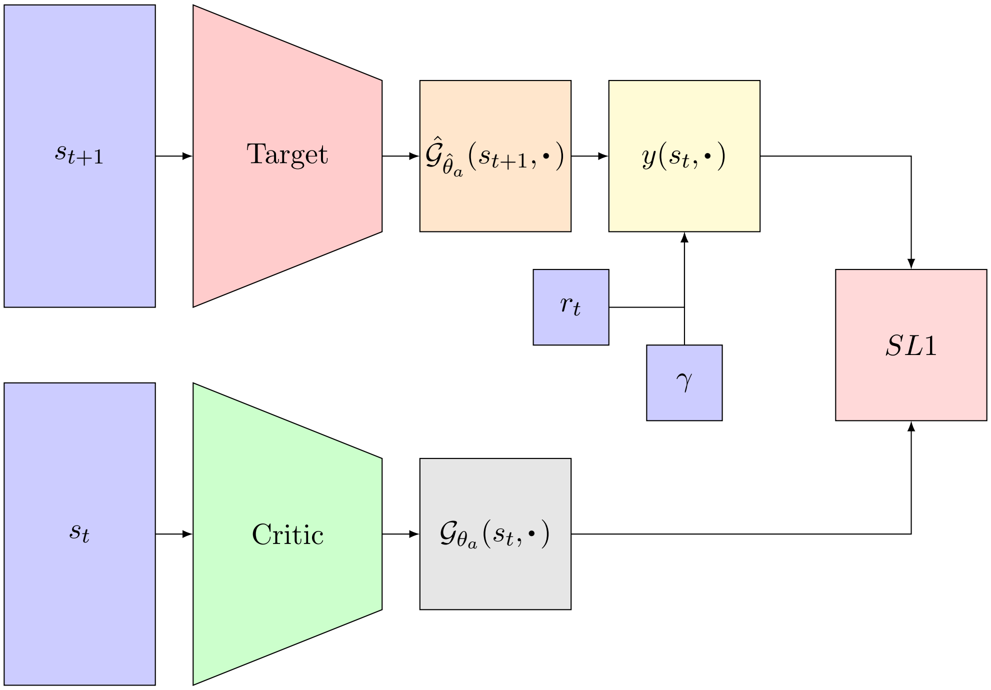

s_{t+1}
Target
$\hat{\mathcal{G}}_{\hat{\theta}_a}(s_{t+1}, \bullet)$
$y(s_t, \bullet)$
r_t
γ
$SL1$
s_t
Critic
$\mathcal{G}_{\theta_a}(s_t, \bullet)$

Figure 12: This figure illustrates the computation of the critic's loss function when the critic is only maximising reward, i.e., when Equation 18 is used for the expected free energy. Briefly, the state s_t is fed into the Critic, and the state s_{t+1} is fed into the target network. The critic outputs the G-values for each action at time t, and the target network outputs the G-values for each action at time $t+1$. Then, the reward, the discount factor, and G-values of each action at time $t+1$ are used to compute the target values $y(s_t, \bullet)$. Finally, the goal is to minimise the SL1 between the prediction of the critic and the target values by changing the weights of the critic.

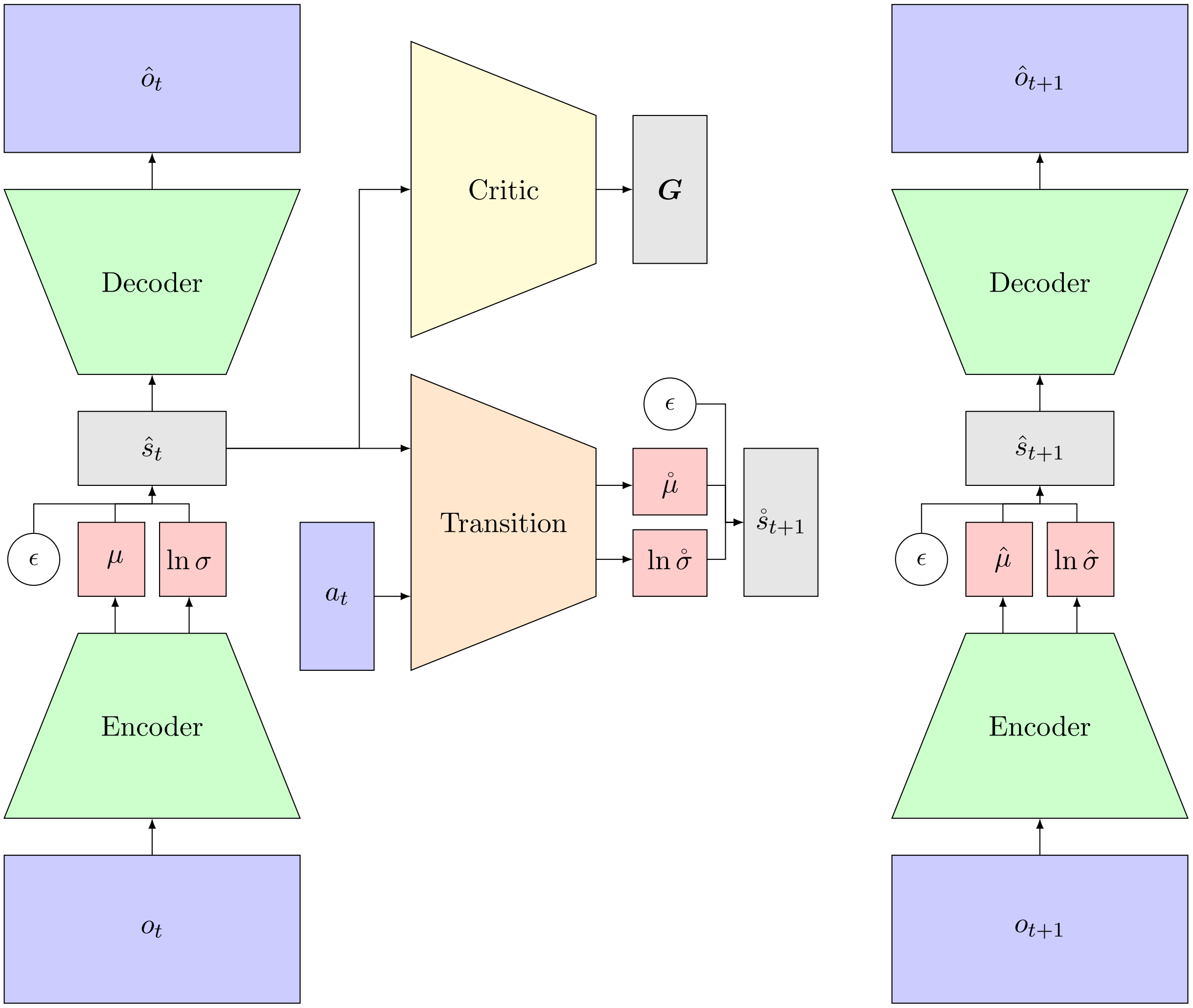


Figure 11: This figure illustrates the CHMM agent. The only new part is the critic network, which takes as input the hidden state at time t and ouputs the expected free energy of each action $\boldsymbol{G}$. Importantly, the CHMM takes actions based on the EFE.

3.5.1 Expected free energy

In this section, we discuss the definition of the expected free energy (EFE) before investigating various ways to implement it in the context of deep active inference. Recently (Parr and Friston, 2018), the expected free energy was defined as:

$$G(\pi) = \sum_{\tau=t+1}^{T} G_\tau(\pi) = \sum_{\tau=t+1}^{T} \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)} \big[\ln Q(s_\tau|\pi) - \ln P(o_\tau, s_\tau) \big], \tag{12}$$

where $P(o_\tau|s_\tau)$ is the likelihood mapping, $Q(s_\tau|\pi)$ is the variational distribution, and ~~literally~~ in the literature $P(o_\tau, s_\tau)$ is called the generative model but is better understood as a target distribution encoding the prior preferences of the agent. Indeed, assuming the standard generative model of active inference (i.e., a partially observable Markov decision process), the hidden states s_τ should depend on $s_{\tau-1}$ and $a_{\tau-1}$. ~~This point is important because it impacts the question of whether $P(o_\tau, s_\tau)$ is indeed the generative model, and therefore whether the expected free energy is the expectation of the variational free energy.~~ **This point is important because it impacts the question of whether $P(o_\tau, s_\tau)$ is indeed the generative model, and therefore whether — as the name sugguest — the expected free energy is the expectation of the variational free energy. More importantly, according to the free energy principle, an active inference agent is supposed to minimise its (variational) free energy. However, if** no relationship can be established between the expected and variational free energy, then one cannot claim that an agent minimising expected free energy also minimises its variational free energy, i.e., one cannot claim that an active inference agent behave according to the free energy principle. Additionally, we need to re-arrange the definition of the EFE stated in (12) to allow rewards to be incorporated:

$$\begin{aligned} G_\tau(\pi) &= \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}\big[\ln Q(s_\tau|\pi) - \ln P(o_\tau, s_\tau)\big] \\ &= \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}\big[\ln Q(s_\tau|\pi) - \ln P(s_\tau|o_\tau) - \ln P(o_\tau)\big] \\ &\approx \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}\big[\ln Q(s_\tau|\pi) - \ln Q(s_\tau) - \ln P(o_\tau)\big] \\ &= \underbrace{D_{\mathrm{KL}}\left[\,Q(s_\tau|\pi)||\,Q(s_\tau)\right]}_{\text{epistemic value}} - \underbrace{\mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}\big[\ln P(o_\tau)\big]}_{\text{extrinsic value}}, \end{aligned} \tag{13}$$

3.5.2 A principled estimate of the EFE at time $t + 1$?

Now, the question is how to estimate (13), and we focus on the case where $\tau = t + 1$. Note, because $\tau = t + 1$, the policy π contains only one action a_t, i.e., $\pi = a_t$. In the tabular version of active inference, the variational distribution is composed of a factor $Q(s_\tau|\pi)$. However, in the deep active inference literature, the variational distribution does not contain such a factor. Generally, a Monte-Carlo estimate is used as follows:

$$Q(s_{t+1}|a_t) = \mathbb{E}_{Q_{\phi_s}(s_t)}\big[P_{\theta_s}(s_{t+1}|s_t, a_t)\big] \approx \frac{1}{N}\sum_{i=1}^{N} P_{\theta_s}(s_{t+1}|s_t = \hat{s}_t^i, a_t), \tag{14}$$

where $\hat{s}_t^i \sim Q_{\phi_s}(s_t)$. Importantly, for the expected free energy to be the expectation of the variational free energy, $Q(s_{t+1}|a_t)$ should be a factor of the variational distribution. However, (14) is estimated using a factor of the generative model $P_{\theta_s}(s_{t+1}|s_t = \hat{s}_t^i, a_t)$. This is a conceptual issue, associated with current deep active inference approaches such as Fountas et al. (2020). In what follows, we use $N = 1$ leading to a simplified version of the estimate:

$$Q(s_{t+1}|a_t) \approx P_{\theta_s}(s_{t+1}|s_t = \hat{s}_t, a_t). \tag{15}$$

Note, in the above equation, $\hat{s}_t^i$ is denoted $\hat{s}_t$ because there is only one sample, i.e., $N = 1$. At this point, we have an estimate for $Q(s_{t+1}|a_t)$ and $Q_{\phi_s}(s_t)$ is the variational distribution. The only missing piece is an estimate of the extrinsic value. In the tabular version of active inference, the preferences of the agent can be related to the rewards from the reinforcement learning literature. In this paper, we follow (Costa et al., 2020b) and define the prior preferences as:

$$P(o_\tau) = \frac{\exp(\psi r_\tau[o_\tau])}{\sum_{o_\tau}\exp(\psi r_\tau[o_\tau])},$$

where ψ is the precision of the prior preferences, and $r_\tau[o_\tau]$ is the reward obtained when making observation o_τ. Taking the logarithm of the above equation leads to:

$$\begin{aligned} \ln P(o_\tau) &= \psi r_\tau[o_\tau] - \ln\sum_{o_\tau}\exp(\psi r_\tau[o_\tau]) \\ &= \psi r_\tau[o_\tau] + C, \end{aligned} \tag{16}$$

where we used the fact that the summation over all o_τ is a normalisation term, i.e., a constant. Using (16), we can now create an estimate of the extrinsic value as follows:

$$\mathbb{E}_{P_{\theta_o}(o_\tau|s_\tau)Q(s_\tau|a_t)}\big[\ln P(o_\tau)\big] \approx \frac{1}{M}\sum_{i=1}^{M}\ln P(o_\tau = \hat{o}_\tau) = \frac{1}{M}\sum_{i=1}^{M}\psi r_\tau[o_\tau] + C,$$

where $\hat{o}_\tau \sim P_{\theta_o}(o_\tau|s_\tau = \hat{s}_\tau)$ and $\hat{s}_\tau \sim Q(s_\tau|a_t)$. In what follows, we use $M = 1$ and discard the constant[6], which leads to a simplified version of the estimate:

$$\mathbb{E}_{P_{\theta_o}(o_\tau|s_\tau)Q(s_\tau|a_t)}\big[\ln P(o_\tau)\big] \triangleq \psi r_\tau[o_\tau] \triangleq \psi r_\tau,$$

[6] Removing a constant does not influence which policy is the best. Indeed, $\pi^* = \arg\max_\pi G(\pi) = \arg\max_\pi G(\pi) - C$.

where we simplied the notation by denoting $r_\tau[o_\tau]$ as r_τ. To conclude, we have the following estimate for the EFE at time $\tau = t + 1$:

$$\begin{aligned} G_{t+1}(a_t) &\approx D_{\text{KL}}\left[\, Q(s_{t+1}|a_t) || \, Q_{\phi_s}(s_{t+1})\right] - \mathbb{E}_{P_{\theta_o}(o_{t+1}|s_{t+1})Q(s_{t+1}|a_t)}\left[\, \ln P(o_{t+1})\right] \\ &\approx D_{\text{KL}}\left[\, P_{\theta_s}(s_{t+1}|s_t = \hat{s}_t, a_t) || \, Q_{\phi_s}(s_{t+1})\right] - \psi r_{t+1}, \end{aligned} \tag{17}$$

where $\hat{s}_t \sim Q_{\phi_s}(s_t)$, $P_{\theta_s}(s_{t+1}|s_t, a_t)$ is known from the generative model, $Q_{\phi_s}(s_{t+1})$ is known from the variational distribution, the KL-divergence can be estimated using an analytical solution, ψ is a hyperparameter modulating the precision of the prior preferences, and r_{t+1} is the reward obtained at time step $t + 1$. As shown in Figure 12, the reward at time step $t + 1$ is used to compute the target values that must be predicted by the critic network.

3.5.3 Other definitions of the EFE at time $t + 1$

In the previous section, we have presented what may be a principled way to estimate the EFE. As will be discussed later in this paper, this estimate of the EFE was not very fruitful empirically. To explore the range of alternatives, we also experimented with the following definitions, which contain relatively minor perturbations of the epistemic value term:

$$\begin{aligned} G^1_{t+1}(a_t) &= H[Q_{\phi_s}(s_{t+1})] - H[P_{\theta_s}(s_{t+1}|s_t = \hat{s}_t, a_t)] - \psi r_{t+1}, \\ G^2_{t+1}(a_t) &= H[P_{\theta_s}(s_{t+1}|s_t = \hat{s}_t, a_t)] - H[Q_{\phi_s}(s_{t+1})] - \psi r_{t+1}, \\ G^3_{t+1}(a_t) &= D_{\text{KL}}\left[\, Q_{\phi_s}(s_{t+1}) || \, P_{\theta_s}(s_{t+1}|s_t = \hat{s}_t, a_t)\right] - \psi r_{t+1}, \end{aligned}$$

where all the entropy terms and the KL-divergence were computed analytically. Also, we experimented with simply predicting the (negative) expected future reward as follows:

$$G^4_{t+1}(a_t) = -\psi r_{t+1},$$

$$G^4_{t+1}(a_t) = -\psi r_{t+1}, \tag{18}$$

which is effectively making the job of the critic identical to the job of the Q-network in the DQN agent (c.f. Section 2.1 for details). More precisely, they are identical, except for the fact that the Q-network is taking observations as input, while the critic takes hidden states.

3.6 Deep Active Inference

In this section, we discuss the full deep active inference (DAI) agent illustrated in Figure 13. Put simply, this agent is composed of five deep neural networks, i.e., the encoder, the decoder, the transition, the critic and the policy network. The decoder $\mathcal{D}_{\theta_o}$ models $P_{\theta_o}(o_\tau|s_\tau)$ as a product of Bernoulli distributions, i.e., $P_{\theta_o}(o_\tau|s_\tau) = \mathcal{B}\text{ernoulli}(o_\tau; \hat{o}_\tau)$ where $\hat{o}_\tau = \mathcal{D}_{\theta_o}(s_\tau)$. The transition network $\mathcal{T}_{\theta_s}$ models $P_{\theta_s}(s_{\tau+1}|s_\tau, a_\tau)$ as a Gaussian distribution, i.e., $P_{\theta_s}(s_{\tau+1}|s_\tau, a_\tau) = \mathcal{N}(s_{\tau+1}|\mathring{\mu}, \mathring{\sigma})$ where $\mathring{\mu}, \ln\mathring{\sigma} = \mathcal{T}_{\theta_s}(s_\tau, a_\tau)$. The critic $\mathcal{G}_{\theta_a}$ outputs a vector containing the predicted expected free energy of each action, which is used to define the prior over action as $P_{\theta_a}(a_\tau|s_\tau) = \sigma[-\zeta\mathcal{G}_{\theta_a}(s_\tau, \bullet)]$, where $\sigma[\bullet]$ is a softmax function and ζ is the precision of the prior over actions. With this in mind, the full generative model of the agent is:

$$P_\theta(o_{o:T}, s_{o:T}, a_{o:T-1}) = P(s_0)\prod_{\tau=0}^{T} P_{\theta_o}(o_\tau|s_\tau)\prod_{\tau=0}^{T-1} P_{\theta_s}(s_{\tau+1}|s_\tau, a_\tau)P_{\theta_a}(a_\tau|s_\tau),$$

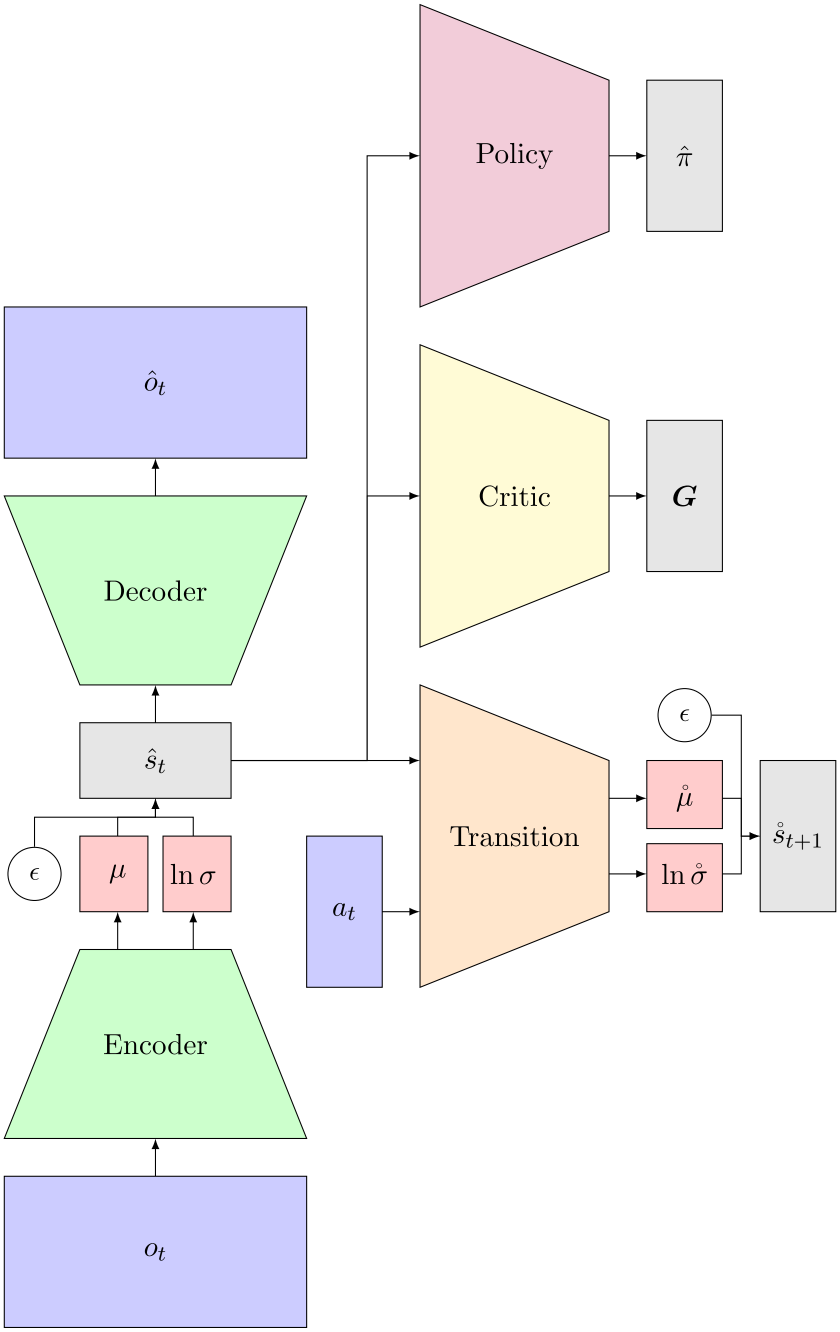


Figure 13: This figure illustrates the DAI agent. The only new part is the policy network, which takes as input the hidden state at time t and ouputs the parameters $\hat{\pi}$ of the variational posterior over actions. Importantly, the DAI takes actions based on the EFE.

where $P(s_0) = \mathcal{N}(s_0; \mu_0, \sigma_0)$ is a Gaussian prior over initial hidden states. Let t be the present time step. The DAI agent maintains posterior beliefs over the present states s_t and action a_t. The variational posterior over states $Q_{\phi_s}(s_t)$ is a Gaussian distribution modelled by the encoder $\mathcal{E}_{\phi_s}$, i.e., $Q_{\phi_s}(s_t) = \mathcal{N}(s_t; \mu, \sigma)$ where $\mu, \ln \sigma = \mathcal{E}_{\phi_s}(o_t)$. The variational posterior over actions $Q_{\phi_a}(a_t|s_t)$ is a categorical distribution modelled by the policy network $\mathcal{P}_{\phi_a}$, i.e., $Q_{\phi_a}(a_t|s_t) = \text{Cat}(a_t; \hat{\pi})$ where $\hat{\pi} = \mathcal{P}_{\phi_a}(s_t)$. The full variational distribution is therefore defined as:

$$Q_\phi(a_t, s_t) = Q_{\phi_a}(a_t|s_t) Q_{\phi_s}(s_t).$$

The variational free energy of the DAI agent is derived in a similar way to the VFE of Section 2.2.2, and is defined as:

$$\begin{aligned} Q^*_\phi(s_t, a_t) &= \underset{Q_\phi(s_t,a_t)}{\arg\min}\, \underbrace{D_{\mathrm{KL}}\left[\, Q_\phi(s_t, a_t) || \, P_{\theta_o}(o_t|s_t) P_{\theta_s}(s_t|s_{t-1}, a_{t-1}) P_{\theta_a}(a_t|s_t)\right]}_{\text{variational free energy}} \\ &= \underset{Q_\phi(s_t,a_t)}{\arg\min} \quad \mathbb{E}_{Q_{\phi_s}(s_t)}\left[D_{\mathrm{KL}}\left[\, Q_{\phi_a}(a_t|s_t) || \, P_{\theta_a}(a_t|s_t)\right]\,\right] + D_{\mathrm{KL}}\left[\, Q_{\phi_s}(s_t) || \, P_{\theta_s}(s_t|s_{t-1}, a_{t-1})\right] \\ &\qquad\qquad - \mathbb{E}_{Q_{\phi_s}(s_t)}[\ln P_{\theta_o}(o_t|s_t)]. \end{aligned} \tag{19}$$

The VFE is therefore a function of s_{t-1}, a_{t-1}, and o_t. Both a_{t-1}, and o_t can be obtained from the replay buffer, and s_{t-1} can be sampled from the variational distribution predicted by the encoder network when observation o_{t-1} is provided as input. Also, the KL-divergences can be computed analytically, the expectations w.r.t $Q_{\phi_s}(s_t)$ can be approximated using a Monte-Carlo estimate, and the logarithm of the likelihood mapping reduces to the ~~BCE~~ binary cross entropy because $P_{\theta_o}(o_t|s_t)$ is a product of Bernoulli distributions. Thus, all the VFE terms can be estimated, and the encoder, decoder, transition and policy networks can be trained to minimise the VFE using gradient descent. The critic's weights are optimised as in Section 3.5 using gradient descent to minimise the smooth L1 norm between the critic's output $\mathcal{G}_{\theta_a}(s_t, \bullet)$ and the target G-values $y(\bullet)$, i.e., $\text{SL1}[\mathcal{G}_{\theta_a}(s_t, \bullet), y(\bullet)]$, where the target G-values are defined as:

$$y(a_t) = G_{t+1}(a_t) + \gamma \mathbb{E}_{Q_{\phi_s}(s_{t+1})}\left[\max_{a_{t+1} \in \mathcal{A}} \hat{\mathcal{G}}_{\hat{\theta}_a}(s_{t+1}, a_{t+1})\right],$$

where $Q_{\phi_s}(s_{t+1})$ can be computed by feeding the image o_{t+1} sampled from the ~~queue~~ replay buffer as input to the encoder, γ is a discount factor, and $G_{t+1}(a_t)$ is the expected free energy received at time $t + 1$ after taking action a_t, i.e.,

$$G_{t+1}(a_t) \approx D_{\mathrm{KL}}\left[\, P_{\theta_s}(s_{t+1}|s_t = \hat{s}_t, a_t) || \, Q_{\phi_s}(s_{t+1})\right] - \psi r_{t+1}, \tag{20}$$

where $\hat{s}_t \sim Q_{\phi_s}(s_t)$, ψ is a hyperparamter modulating the precision of the prior preferences, and r_{t+1} is the reward obtained at time step $t + 1$. Note, we also experimented with other definitions of the EFE at time $t + 1$ as presented in Section 3.5.3. Finally, with regard to the action selection performed by the DAI agent, there are at least four possibilities: (i) select a random action, (ii) select the action with the highest posterior probability according to the policy network, i.e., $a^*_t = \arg\max_{a_t} Q_{\phi_a}(a_t|s_t)$, (iii) sample an action from the posterior over actions, i.e., $a^*_t \sim Q_{\phi_a}(a_t|s_t)$, and (iv) use the $\mathring{\epsilon}$-greedy algorithm with exponential decay, i.e., random action with probabilty $\mathring{\epsilon}$ or best action with probability $1 - \mathring{\epsilon}$.

4. Results

In this section, we discuss the results obtained by the DQN agent and each model presented in Section ~~2~~ 3 at solving the dSprites problem. The code that can be used to reproduce all the experiements can be found on GitHub at the following URL: `https://github.com/ChampiB/Challenges_Deep_Active_Inference`. Section 4.1 presents the results obtained by the DQN agent. Section 4.2 presents the VFE obtained by the VAE agent, and shows the reconstructed images produced by the VAE. Section 4.3 shows the VFE of the HMM agent as well as the generated sequences of images. Section 4.4 illustrates the VFE obtained by the CHMM as well as the reward obtained by this model when using different action selection schemes and different definitions for the EFE. Finally, Section 4.5 dicusses the VFE obtained by the DAI agent, as well as the rewards obtained by this model. Note, each time CKA is used in the following sections, we sampled 5K data examples, and we used them to compute all the CKA scores.

4.1 DQN agent

In this section, we report the results obtained from the DQN agent. As shown in Figure 14, the DQN was able to accumulate a total amount of reward of around 50K. This result confirms the correctness of our implementation, and gives us a baseline which can be used to evaluate the performance of the CHMM and DAI agents. To better understand the representations learned by a DQN, we compute the CKA scores between the activations of its layers.

We can see in Figure 15 that while the layers closer to the input retain some similarity with each other, the last two layers learn highly specific representations.

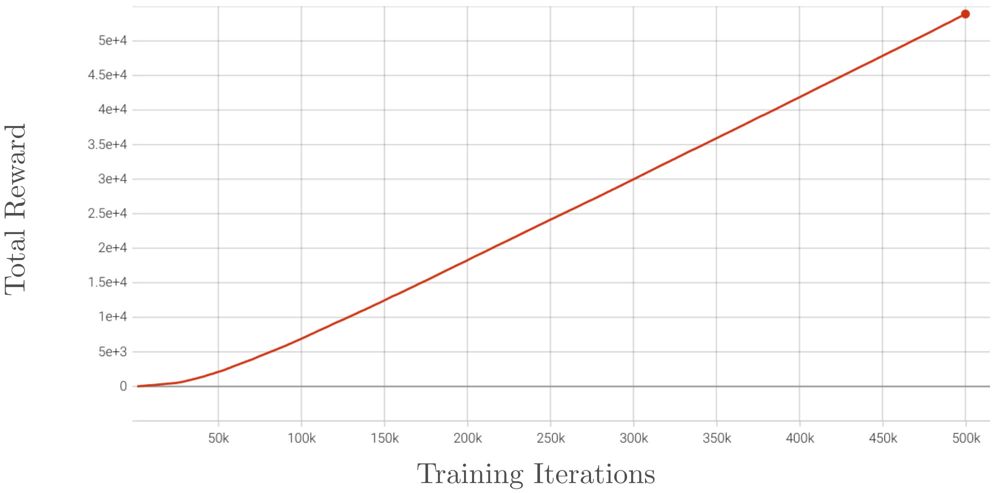


Figure 14: This figure illustrates the cumulated rewards obtained by the DQN agent during the 500K training iterations.

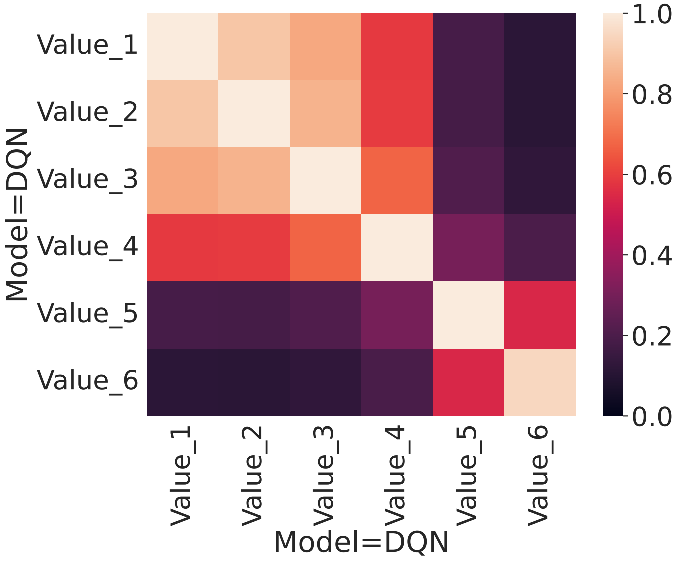


Figure 15: Value_X labels the X-th layer of the value network, i.e., Value_1 is the closest to the input and Value_6 is the output layer. We can see that the first three layers of the DQN learn very similar representations (CKA is close to 1). The representations learned by the fourth layer start to diverge (CKA is lower), and the last two layers learn highly specific representations that are very different from the previous layers (CKA is close to 0), but slightly similar to each other.

4.2 VAE agent

In this section, we report the results obtained by the VAE agent. As shown in Figure 16, the VFE decreases as training progresses. Also, at the end of the 500K training iterations, the VAE is able to properly reconstruct the images, c.f., Figure 17. Additionally, since the VAE takes random actions in the environment, the agent was unable to solve the task and accumulated a total amount of reward of around -7K. Those results suggest our implementation is correct, and gives us a baseline for the amount of rewards obtained under random play in the dSprites environment.

To observe the representations learned by VAEs, we compute the CKA scores between the activations of the different layers of the encoder. We can see in Figure 18a that the representations are strongly similar between all layers, with the exception of the mean and variance representations at the output end (last two layers), similarly to what was observed in Bonheme and Grzes (2022), and is therefore expected. As illustrated in Figure 18b, these representations are generally similar to those learned by the DQN in early layers but the representations of the last two layers strongly differ, reflecting the difference of learning objectives between the VAE and DQN.

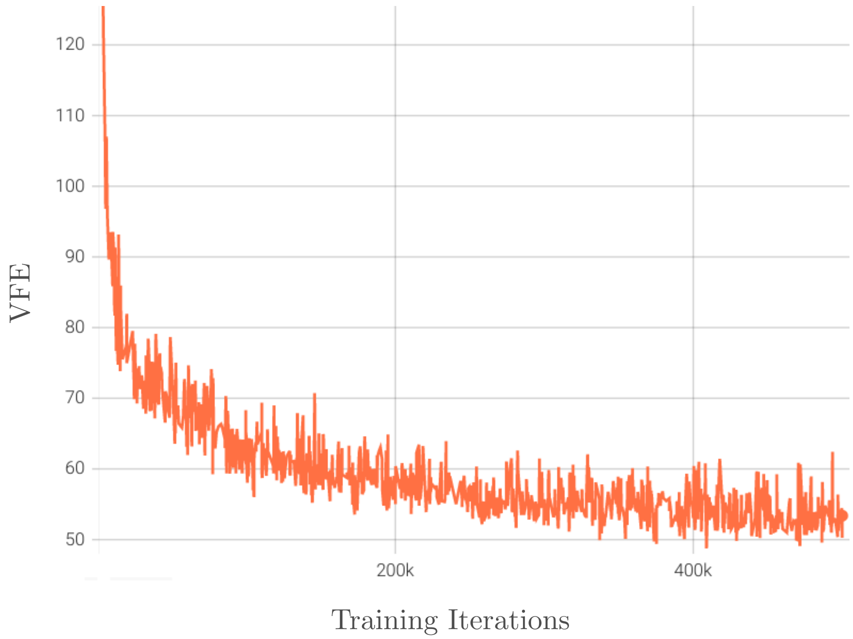


Figure 16: This figure illustrates the variational free energy of the VAE agent during the 500K iterations of training.

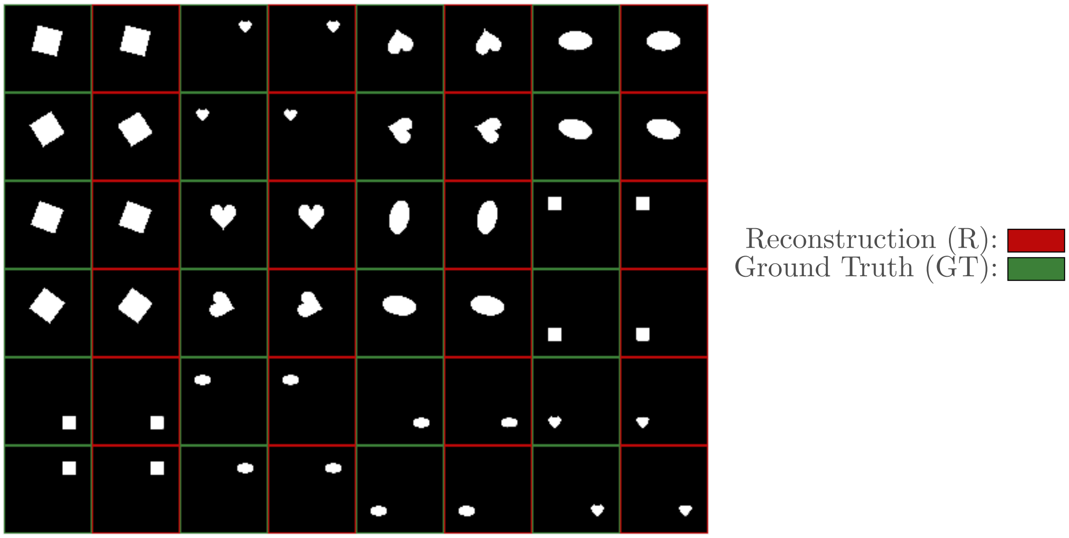
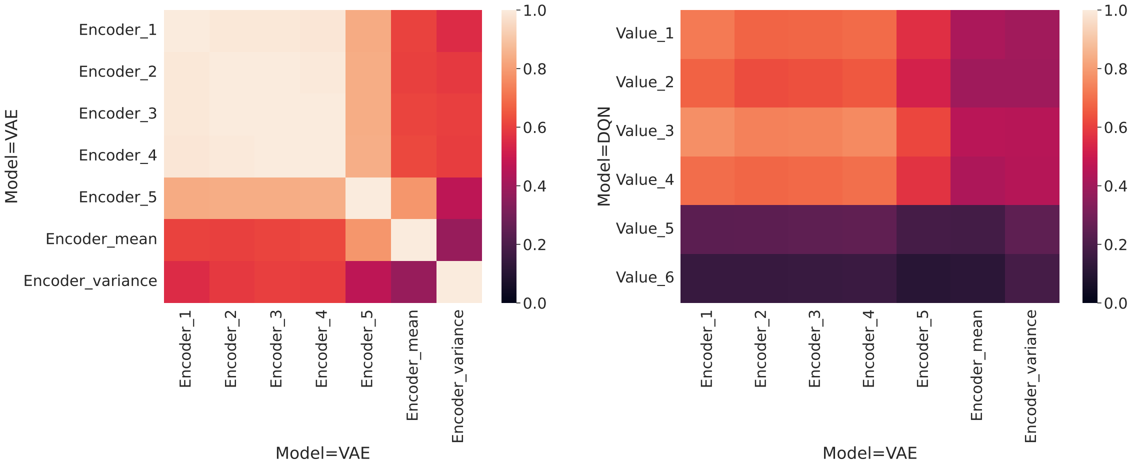


Figure 17: This figure illustrates the reconstructed image produced by the VAE after 500K training iterations. The columns alternate between the input images and the reconstructed images.



(a)

(b)

Figure 18: Encoder_X is to the X-th layer of the encoder network, i.e., Encoder_1 is the closest to the input and Encoder_5 is the one just before the mean and variance layers. Encoder_mean and Encoder_variance are the mean and variance layers of the encoder network, respectively. (a) shows the similarity between the representations learned by different layers of the encoder of a VAE. (b) shows the similarity between the representations learned by a DQN and a VAE. Note, both the VAE and DQN take images as input and need to process them to either learn a compact representation and reconstruct the images or predict the cumulated reward, respectively. Thus, both learn to represent edges and combination of edges in their first layers.

4.3 HMM agent

In this section, we report the results obtained by the HMM agent. As shown in Figure 19, the VFE decreases as training progresses. By comparing Figure 16 and 19, one can see that the HMM has a lower VFE than the VAE. This is because the agent has more flexibility regarding the prior, i.e., the log-likelihood is the same between the two models but the complexity term is smaller for the HMM than for the VAE. Also, at the end of the 500K training iterations, the HMM is able to properly generate sequences of images, c.f., Figure 20. Additionally, since the HMM takes random actions in the environment, the agent was unable to solve the task and accumulated a total amount of reward of around -7K. These results suggest that our implementation is correct, and comfirm our baseline for the amount of rewards obtained under random play in the dSprites environment.

Similarly to VAEs, we are interested in observing the representations learned by the encoder of the HMM, ~~but~~ and also by its transition network. The representations learned by the encoder of the HMM follow the same trend as those learned by VAEs with an even more marked dissimilarity between the log variance of the HMM and the other representations learned by this model, as illustrated in Figures 21a and 21b. We can further observe in Figure 21a that the transition network learns representations similar to the mean representation (Encoder_mean of HMM) in its first three layers, while the representations learned by the last layer (Transition_variance) are not similar to any other representations learned by the transition or encoder networks. We can also see in Figure 21b that the mean and variance representations (Encoder_mean and Encoder_variance) learned by HMMs are different from those learned by VAEs, possibly indicating that the transition network influences these two representations. Similarly to VAEs, one can observe in Figure 21c, that the representations learned by the variance layers of the encoder and transition networks (Encoder_variance and Transition_variance) are very different to the representation learned by the DQN. In contrast, the first four layers of the encoder (Encoder_1 to Encoder_4) are similar to the representation learned by the first layers of the DQN (Value_1 to Value_4), but are very different from the last two layers (Value_5 and Value_6).

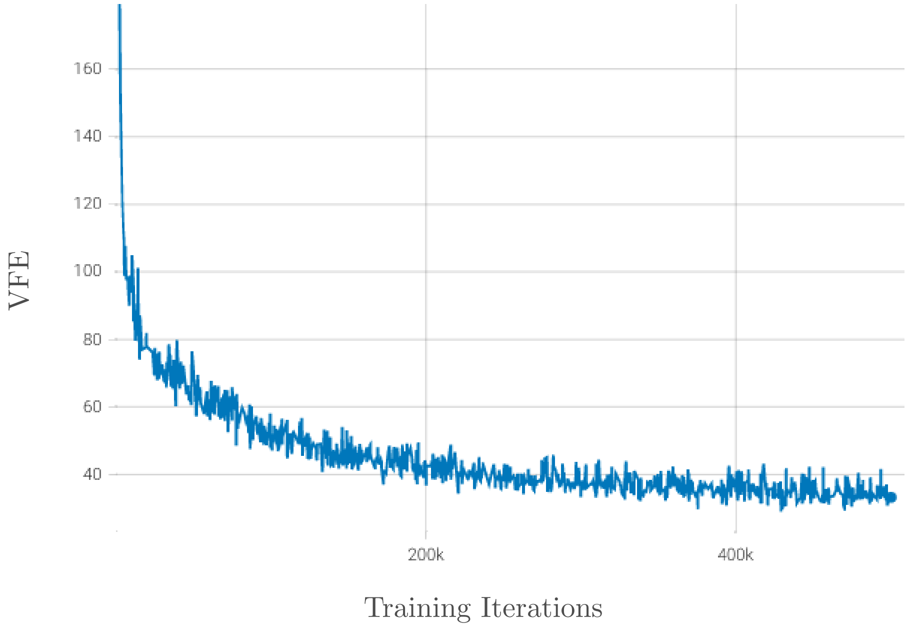


Figure 19: This figure illustrates the variational free energy of the HMM agent during the 500K iterations of training.

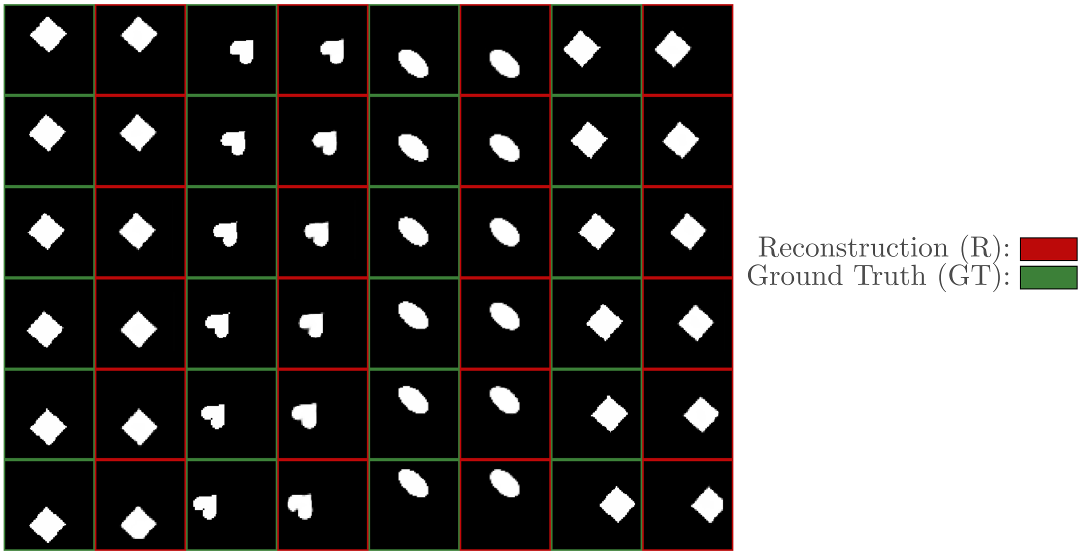
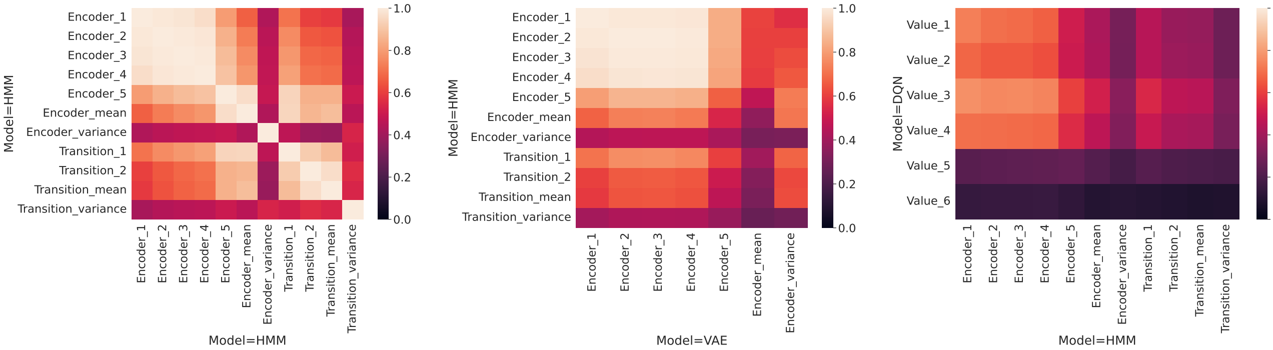


Figure 20: This figure illustrates the sequences of reconstructed images generated by the HMM after 500K training iterations. The columns alternate between the ground truth images and the reconstructed images. Time passes vertically (from top to bottom), and within each column, the same action is executed repeatedly.

4.4 CHMM agent

In this section, we report the results obtained by the CHMM agent, when using different action selection strategies and different definitions of the expected free energy. Figure 22 presents the cumulated rewards obtained by the CHMM agents using an $\mathring{\epsilon}$-greedy algorithm for action selection, as well as the total rewards obtained by the DQN agent. The critic network of the CHMM agents were trained to predict the five definitions of the expected free energy proposed in Section 3.5. Note, the DQN agent performs better that any of the CHMM agents, and the only agent that solves



(a)

(b)

(c)

Figure 21: Transition_X is to the X-th layer of the transition network, i.e., Transition_1 is the closest to the input and Transition_2 is the one just before the mean and variance layers. Transition_mean and Transition_variance are the mean and variance layers of the transition network, respectively. (a) shows the similarity between the representations learned by different layers of the encoder and transition network of an HMM. (b) shows the similarity between the representations learned by an HMM and a VAE. (c) shows the similarity between the representations learned by a DQN and an HMM.

the task is the CHMM ~~only maximising reward~~ maximising reward only, i.e., without any information gain. Figure 23 presents the same experiement as Figure 22 except that the CHMM agents were using softmax sampling for action selection. In this setting, none of the CHMM agents were able to solve the task. Finally, Figure 24 presents yet again the same experiements but this time the CHMM agents were selecting the best action according to the critic. In this setting, only the CHMM maximising reward was able to solve the task. Also, by comparing Figures 22 and 24, it becomes clear that the CHMM using an $\mathring{\epsilon}$-greedy algorithm performs better than the CHMM selecting the best action according to the critic. Put simply, the latter suffers from a lack of exploration that slows down its learning.

Additionally, Figure 25 represents the variational free energy of the CHMM agent using the $\mathring{\epsilon}$-greedy algorithm. All the agents were able to minimise their variational free energy, except the one displayed in orange ~~which crashed~~whose VFE suddenly became equal to NaN (i.e., Not a Number); this agent was minimising the expected free energy as defined by G^1. Note, G^1 is neither the reward nor the "principled" expected free energy, G^1 is one of definitions that we experimented with to explore alternative definitions. Also, the variational free energy of the CHMM agents using softmax sampling and best action selection are not presented, because their results are very similar to the results shown in Figure 25.

Finally, Figure 26 shows examples of predicted trajectories after a CHMM (maximising reward) was trained. By comparing with Figure 20, we see that the CHMM does not understand the dynamics of the environment as well as the HMM agent. This suggests a conflict between the two goals of the agent, i.e., maximising reward[7] (or expected free energy) and learning a model of the world. More precisely, Figure 20 shows that an HMM agent taking random actions is able to gather a large diversity of training examples and learns the dynamic of the environment beautifully, but does not solve the task. In contrast, the CHMM maximising reward solves the task but learns a poor model of the environment, c.f., Figure 26.

[7] As shown in Figure 12, the reward is used to compute the target values that must be predicted by the critic network.

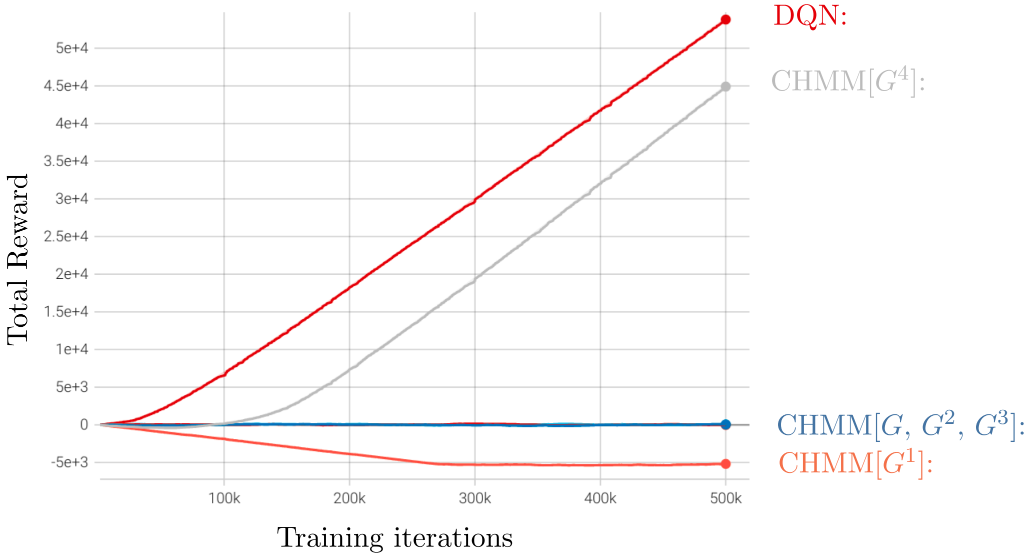


Figure 22: This figure illustrates the total amount of reward gathered by the CHMM agents (with $\mathring{\epsilon}$-greedy action selection) during the 500K iterations of training. The only two models that were able to solve the task are the ones maximising reward (without information gain), i.e., the DQN agent in red and the CHMM whose critic network was predicting only reward in gray.

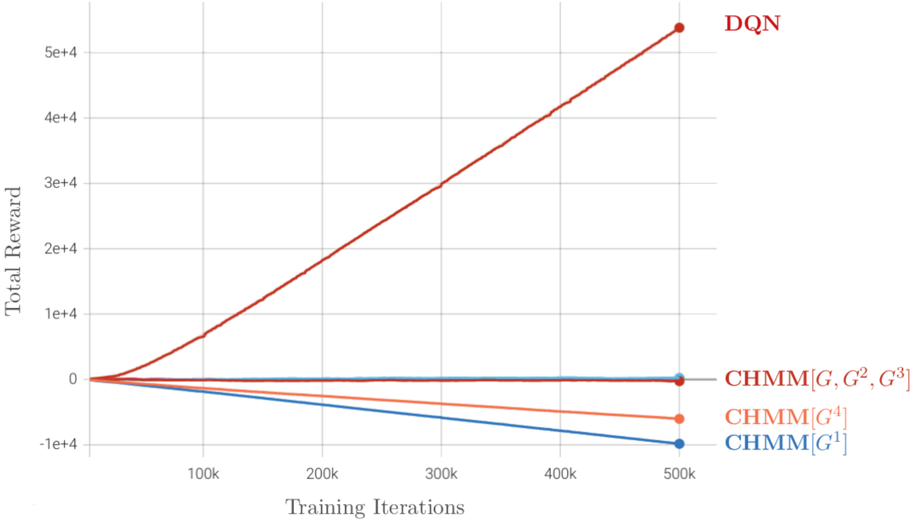


Figure 23: This figure illustrates the total amount of reward gathered by the CHMM agents (with softmax sampling) during the 500K iterations of training. The only model that was able to solve the task is the DQN agent in red, and all CHMM agents failed.

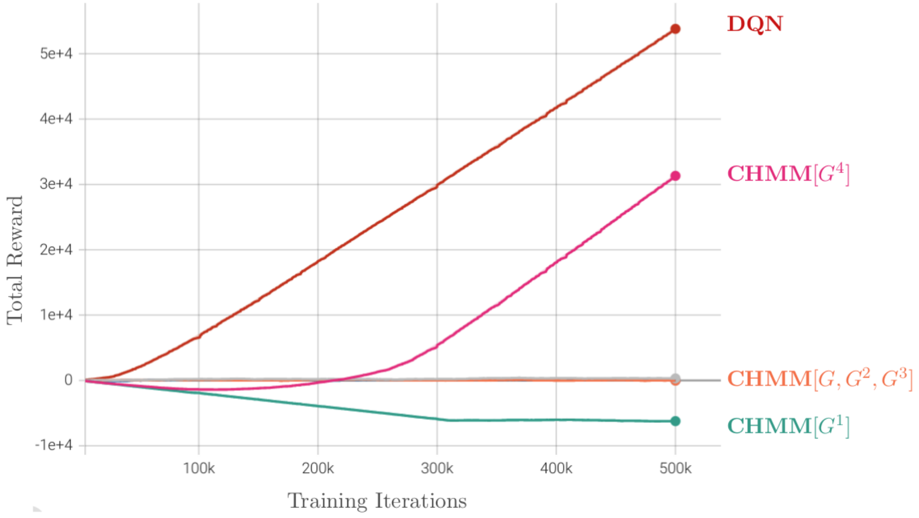


Figure 24: This figure illustrates the total amount of reward gathered by the CHMM agents (with best action selection) during the 500K iterations of training. The only two models that were able to solve the task are the ones maximising reward (without information gain), i.e., the DQN agent in red and the CHMM whose critic network was predicting only reward in pink.

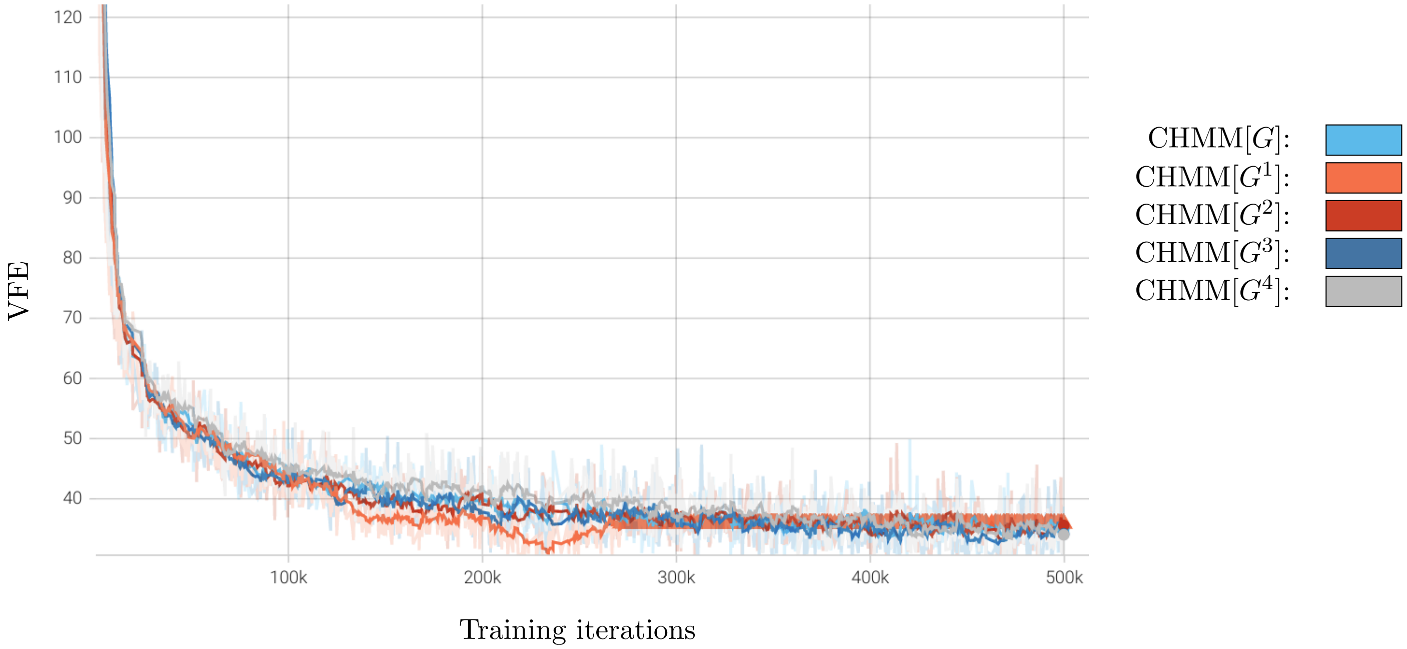


Figure 25: This figure illustrates the variational free energy of the CHMM agents during the 500K iterations of training. All the agents were able to minimise their variational free energy, except the one displayed in orange which crashed; this agent was minimising the expected free energy as defined by G^1. ~~The crash~~ More precisely, the variational free energy took the value “Not a Number” (NaN), which is visible because of the thick horizontal line between 270K and 500K training iterations~~, which corresponds to the variational free energy taking the value “not a number” (NaN)~~.

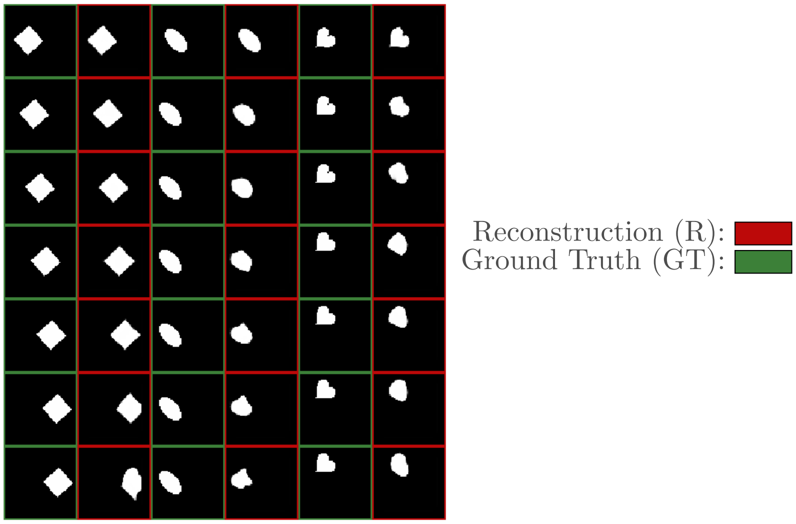
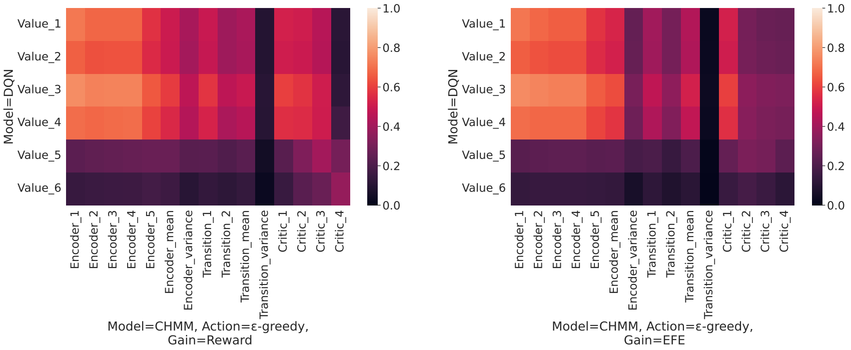


Figure 26: This figure illustrates the sequences of reconstructed images generated by a CHMM (maximising reward) after 500K training iterations. The columns alternate between the ground truth images and the reconstructed images. Time passes vertically (from top to bottom), and within each column, the same action is executed repeatedly.

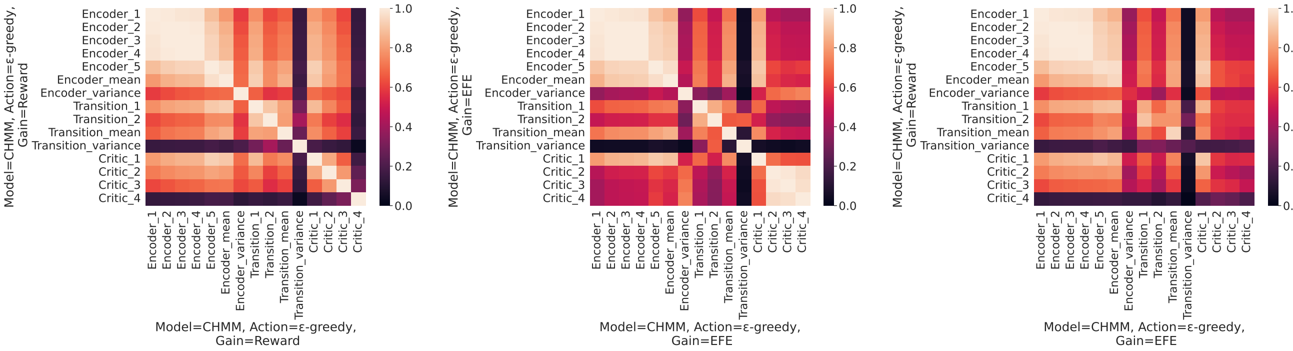
4.4.1 How do CHMMs learn?

CHMM with $\mathring{\epsilon}$-greedy action selection As illustrated in Figure 22, only the CHMM whose critic maximises the reward was able to solve the task. We could thus expect the representations learned by this CHMM to be closer to those learned by the DQN than those learned by the other CHMMs. We can see in Figure 27 that the last two layers of the critic network of the CHMM maximising the reward are indeed a bit more similar to the representations of the last two layers of the DQN than the representations learned when the critic is minimising the EFE (see intersection of Value_5 and Value_6 with Critic_3 and Critic_4, i.e., bottom right corner of the matrix). However, the representations learned by the critic of both CHMMs are still quite different from the last two layers of the DQN (CKA is lower than 0.4, bottom right corner of the matrix, again). Interestingly, the first four layers of the CHMM maximising the reward retain a high similarity with the earlier layers of the DQN (4 × 4 region at upper left), suggesting some common representations between models. We can further see in Figure 28a that the CKA score between the encoder, transition and critic networks is higher or equal to 0.6 (except for the variance layer of the transition and the last layer of the critic), indicating that the transition and critic networks of the CHMM maximising the reward retain some information from the encoder. The information retained by the first three layers of the critic when the CHMM minimises the EFE is much lower, as illustrated in Figures 28b.



(a) (b)

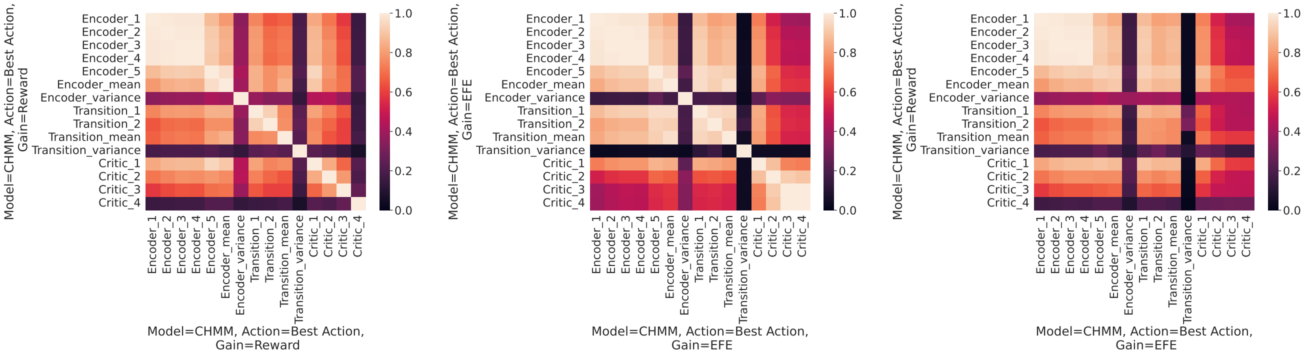
Figure 27: (a) shows the similarity between the representations learned by a CHMM whose critic maximises the reward (with $\mathring{\epsilon}$-greedy selection) and a DQN; (b) shows the similarity between the representations learned by a CHMM whose critic minimises the EFE (with $\mathring{\epsilon}$-greedy selection) and a DQN



(a) (b) (c)

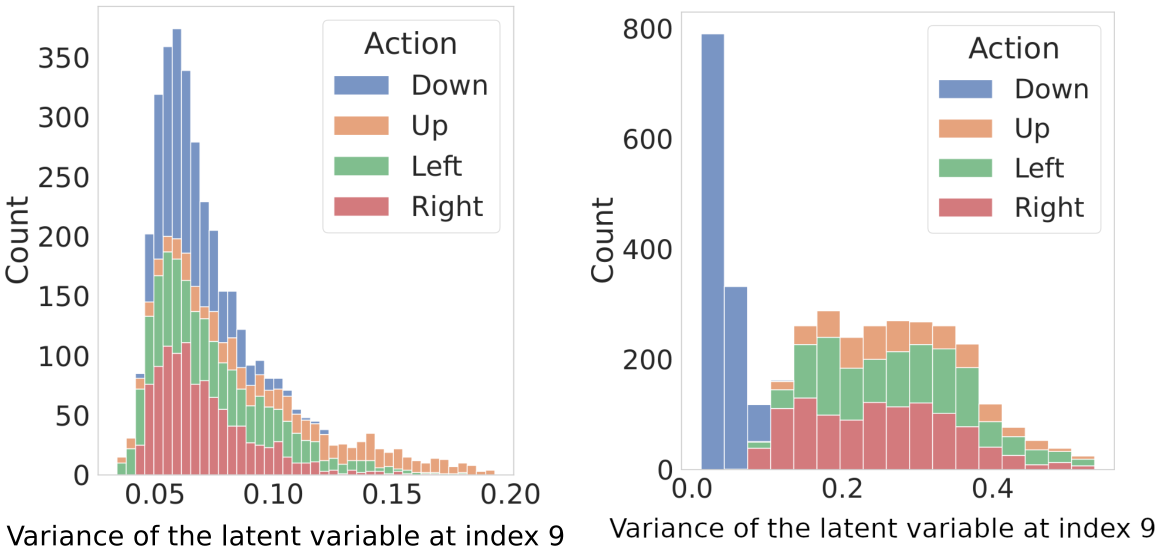
Figure 28: (a) shows the similarity between the representations learned by different layers of the encoder, transition and critic networks of a CHMM whose critic maximises the reward (with $\mathring{\epsilon}$-greedy selection). (b) shows the similarity between the representations learned by different layers of the encoder, transition and critic networks of a CHMM whose critic minimises the EFE (with $\mathring{\epsilon}$-greedy selection) (c) shows the similarity between the representations learned by two CHMMs, one whose critic optimises EFE and the other optimises reward (both with $\mathring{\epsilon}$-greedy selection).

CHMM with best action selection As illustrated in Figure 24, only the CHMM whose critic maximises the reward was able to solve the task. However, one can see in Figure 29 that the CHMM whose critic minimises the EFE learns representations similar to those of the CHMM maximising the reward in most layers, with the exception of the variance layer of the encoder and transition network (Encoder_variance and Transition_variance). To better understand the differences between the representations learned by the variance layer of both models, we fed 5K state-action pairs through the transition network, and displayed the distribution of the variances outputed by the transition network. This analysis reveals that the variance (of the variance layer) of the transition network is very small and does not change much when maximising the reward but is larger and varies more when minimising the EFE as illustrated in Figure 30. This reflects a higher uncertainty of the transitions for the CHMM minimising EFE. More specifically, the CHMM minimising EFE seems to be confident for the action down, but is very uncertain for all the other actions, which suggests that the CHMM minimising EFE always picks the action down and does not gather enough data for the other actions.



(a) (b) (c)

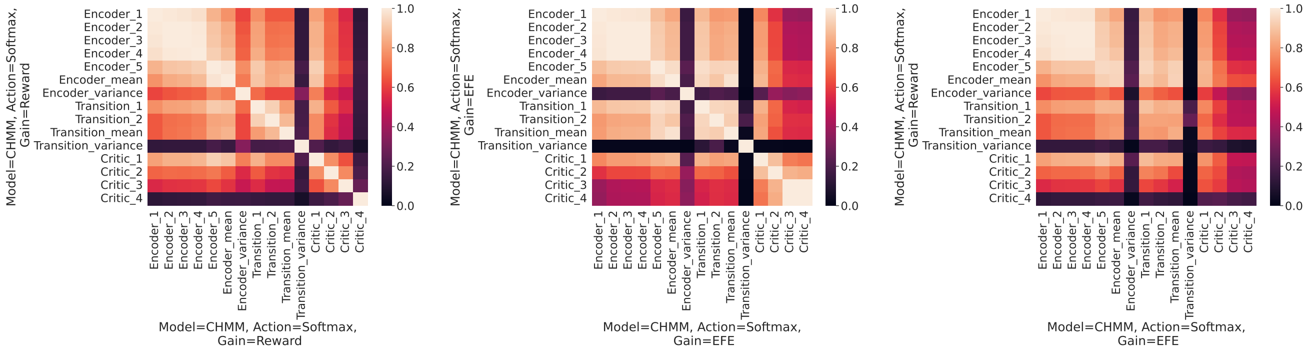
Figure 29: (a) shows the similarity between the representations learned by different layers of the encoder, transition and critic networks of a CHMM whose critic maximises the reward (with best action selection). (b) shows the similarity between the representations learned by different layers of the encoder, transition and critic networks of a CHMM whose critic minimises the EFE (with best action selection) (c) shows the similarity between the representations learned by two CHMMs, one whose critic optimises EFE and the other that optimises reward (both with best action selection).



(a) (b)

Figure 30: (a) shows one latent dimension of the variance layer of the transition network for the CHMM maximising the reward. (b) shows one latent dimension of the variance layer of the transition network for the CHMM minimising the EFE. Both figures are typical of the distributions of variance activations in the two models. Note, only the action down has low variance for the CHMM minimising the EFE. This suggests that the CHMM minimising the EFE always picks the action down, and does not gather enough data for the other actions.

CHMM with softmax action selection As illustrated in Figure 23, none of the CHMMs with softmax action selection were able to solve the task. Once again, the variance (of the variance layer) of the transition network is very different in the CHMM whose critic minimises the EFE compared to the CHMM whose critic maximises the reward (see Figure 31c at the intersection of the two Transition_variances). We further observe the same trend regarding the uncertainty of the output of the transition network when optimising the EFE, as shown in Figure 32. While this may explain why the model optimising the EFE does not solve the task, this does not indicate why the model maximising the reward cannot solve the task, and we can hypothesise that those results may be attributed to the softmax action selection. More precisely, if the values predicted by the critic network are very close to each other, then an agent using softmax sampling may perform random actions.

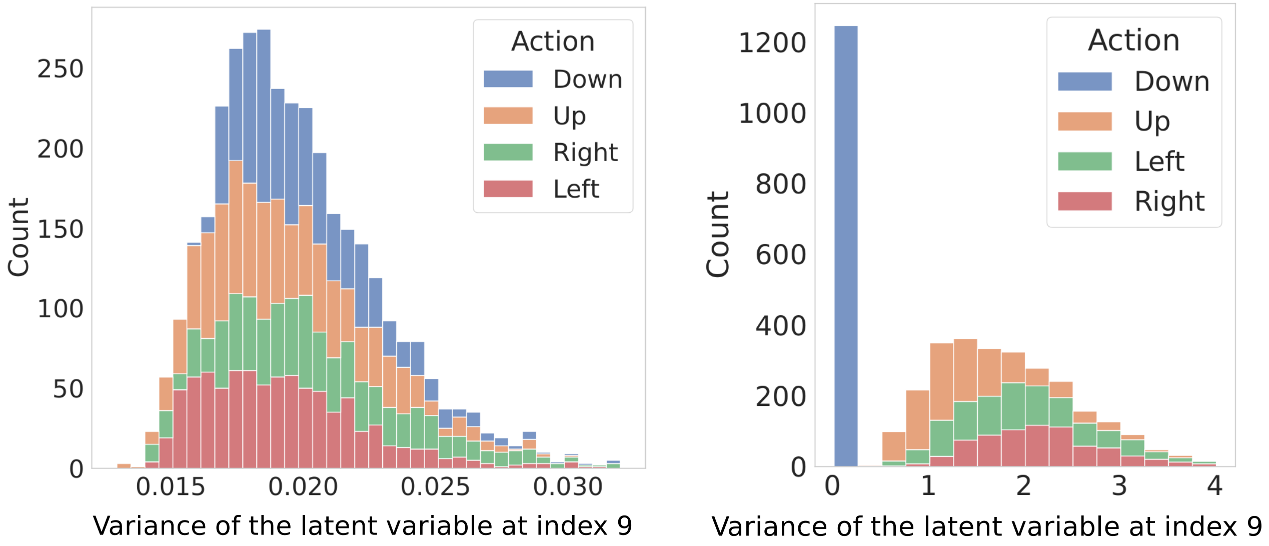


(a)

(b)

(c)

Figure 31: (a) shows the similarity between the representations learned by different layers of the encoder, transition and critic networks of a CHMM whose critic maximises the reward (with softmax action selection). (b) shows the similarity between the representations learned by different layers of the encoder, transition and critic networks of a CHMM whose critic minimises the EFE (with softmax action selection). (c) shows the similarity between the representations learned by two CHMMs, one whose critic optimises EFE and the other that optimises reward (both with softmax action selection).



(a)

(b)

Figure 32: (a) shows one latent dimension of the variance layer of the transition network for the CHMM maximising the reward. (b) shows one latent dimension of the variance layer of the transition network for the CHMM minimising the EFE. Both figures are representative of the distributions of variance activations in the two models. Note, only the action down has low variance for the CHMM minimising the EFE. This suggests that the CHMM minimising the EFE always pick the action down, and does not gather enough data for the other actions.

4.4.2 Degenerate behaviour with the expected free energy?

Up to now, we saw that the CHMM minimising expected free energy (EFE) was not able to solve the task. Also, we discovered that the transition network is uncertain for the actions: up, left, and right, which suggests that the CHMM minimising EFE always takes action down. Figure 33 corroborates this story. Indeed, Figure 33a shows that the agent minimising EFE almost exclusively picked action down, and Figure 33b shows that the entropy of the prior over actions very quickly converges to zero.

In contrast, the CHMM maximising reward, keeps on selecting the actions right and left, which enables it to drag the shape towards the appropriate corner (see Figure 33c). Also, as shown in Figure 33d, the entropy of the prior over actions remained a lot higher than zero.

Note, the only difference between the CHMM minimising EFE and the one maximising reward is the information gain, which is defined as the KL divergence between the output of the transition and encoder networks. Since the EFE is minimised, the output of those two networks need to be as ~~closed~~ close as possible to each other.

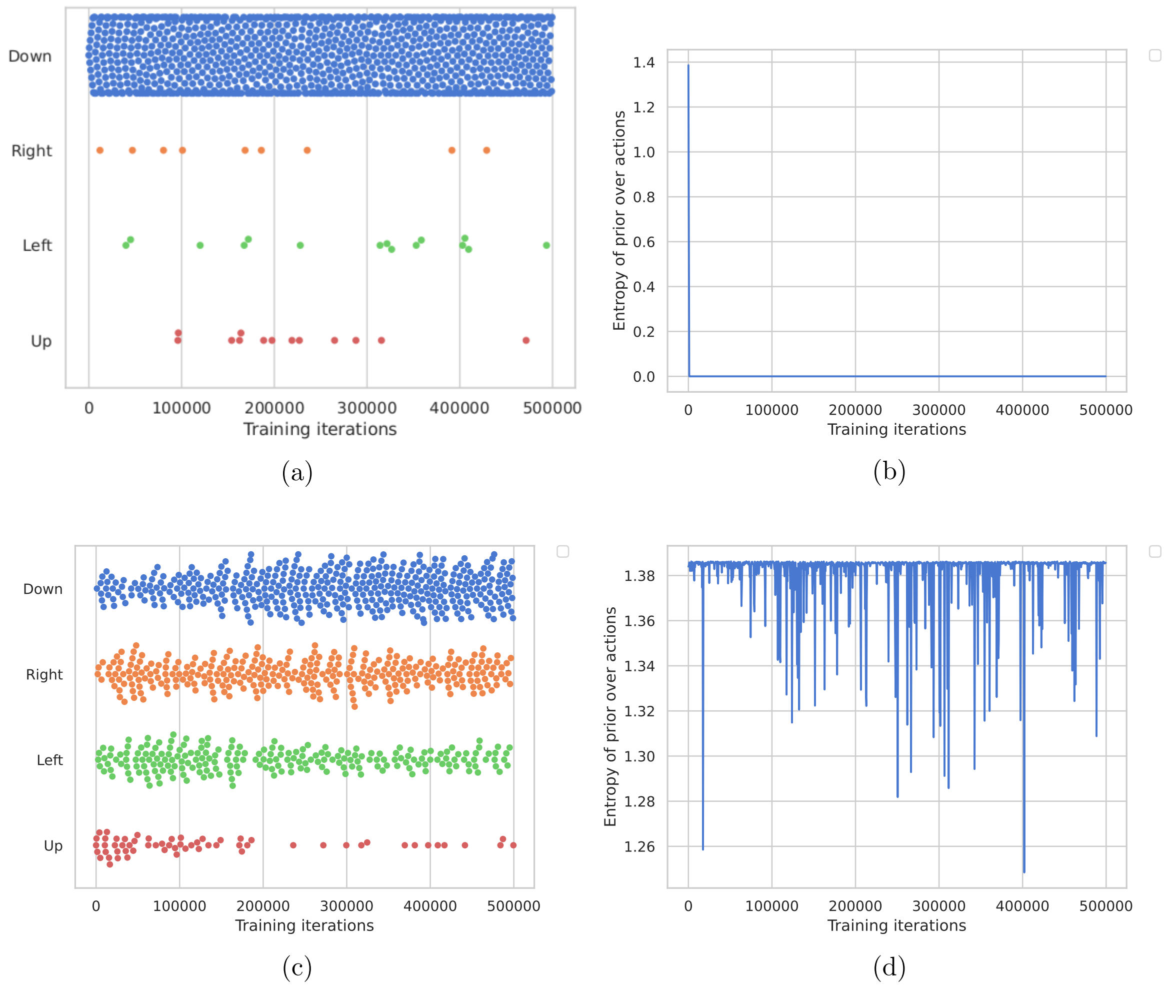


Figure 33: (a) shows the action taken for each planning iteration when the CHMM is minimising expected free energy. (b) shows the entropy of the prior over actions when the CHMM is minimising the EFE. (c) shows the action taken for each planning iteration when the CHMM is maximising reward. (d) shows the entropy of the prior over actions when the CHMM is maximising reward.

This suggests that the CHMM minimising EFE is picking a single action (down), and becomes an expert at predicting the future when selecting this action. This effectively makes the KL divergence between the output of the transition and encoder networks small. Additionally, when selecting the action down, the average reward is zero, because (in the dSprites dataset) there are as many shapes on the left of the image as on the right, and when crossing the bottom line, the agent receives a reward which is linearly increasing (or decreasing) as a corner is approached and is zero at the center of the image. For all the other actions, the expected reward will be negative because after 50 action-perception cycles without crossing the bottom line, the trial is interrupted and the agent receive a reward of -1. Thus, if the CHMM has to stick to a single action to keep the KL divergence small, then the best action it can choose is down, i.e., action down has the highest expected reward.

~~(a) shows the action taken for each planning iteration when the CHMM is minimising expected free energy. (b) shows the entropy of the prior over actions when the CHMM is minimising the EFE. (c) shows the action taken for each planning iteration when the CHMM is maximising reward. (d) shows the entropy of the prior over actions when the CHMM is maximising reward.~~

Also, Figure 34 shows the impact of adding X% of the information gain into the objective function, i.e., the agent starts by only maximising reward (c.f. Equation 18), and after 200K training iterations minimises reward plus X% of the information gain. One can see that adding even 1% of the information gain already dramatically decreases the amount of reward gathered.

~~This figure illustrates the total reward aggregated by CHMM agents during the 500K iterations of training. All the agents start by only maximising reward, and after 200K training iterations, X% of the information gain is added to the objective function. Note, even adding 1% of the information gain is enough to drastically reduce the total reward aggregated by the agent. The differences in trajectories before 200K are arbitrary, arising from differences in random initializations.~~

To conclude, the same information gain that is intended to give an EFE minimising agent its exploration behaviour, also prevents the agent from solving the dSprites environment. This is because the agent is reduced to picking a single action, leading to a suboptimal policy.

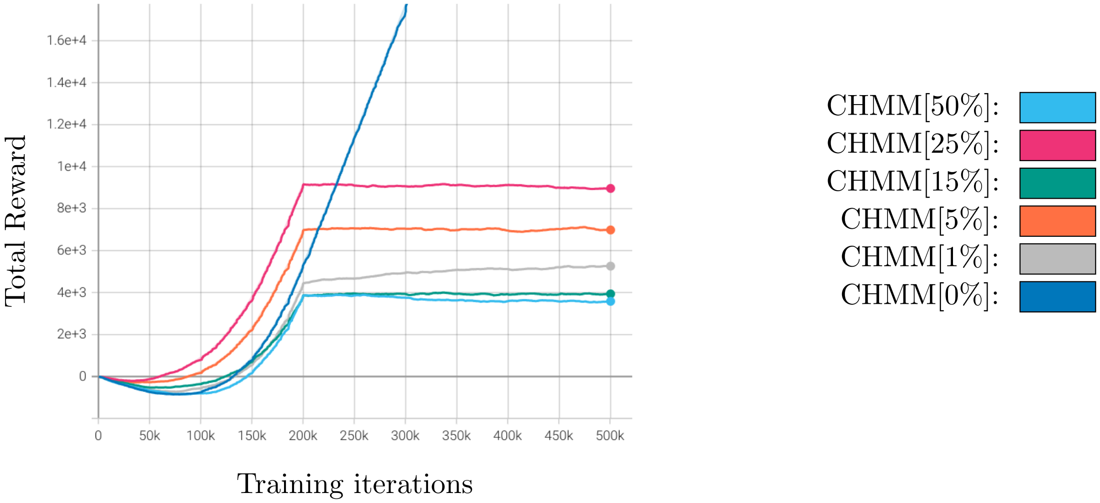


Figure 34: This figure illustrates the total reward aggregated by CHMM agents during the 500K iterations of training. All the agents start by only maximising reward, and after 200K training iterations, X% of the information gain is added to the objective function. Note, even adding 1% of the information gain is enough to drastically reduce the total reward aggregated by the agent. The differences in trajectories before 200K are arbitrary, arising from differences in random initializations.

4.5 DAI agent

In this section, we report the results obtained by the DAI agent, when using different action selection strategies and different definitions of the expected free energy. First, most of the fifteen DAI agents crashed because of numerical instability, i.e., the VFE suddenly became "Not a Number". The only DAI agent that survived (i.e., did not crash) was maximising rewards while performing softmax sampling for action selection. Figure 37 shows that the DAI agent successfully minimises its variational free energy, but as shown in Figure 36, the DAI agent does not solve the task and performs as well as a random agent. Finally, Figure 35 shows sequences of images produced by the DAI agent after 500K training iterations. Note, while the agent does not solve that task, it understands the dynamics of the environment pretty well. However, the agent struggles with images representing hearts.

By comparing Figures 20, 26 and 35, we see that the DAI agent with softmax sampling has a better reconstruction than the CHMM agent with the $\mathring{\epsilon}$-greedy algorithm (which is presented in Figure 26). In contrast, the DAI agent does not reconstruct the sequences of images as well as the HMM agent performing random actions (which is presented in Figure 20).

As previously mentioned, the only DAI that did not crash during training used softmax action selection and had a critic maximising reward. We can see in Figures 38b and 38a that the representational similarity between this DAI and DQN is very close to the representational similarity between a DQN and a CHMM using the same action selection and maximising reward. This is further confirmed by a comparison between the CHMM and the DAI model in Figure 38c. Interestingly, we can see that the policy and critic network learn similar representations, indicating that the policy network is learning correctly. However, we previously inferred that the softmax action selection may be suboptimal and this seems to hold true for the DAI as well, given that it is unable to solve the task.

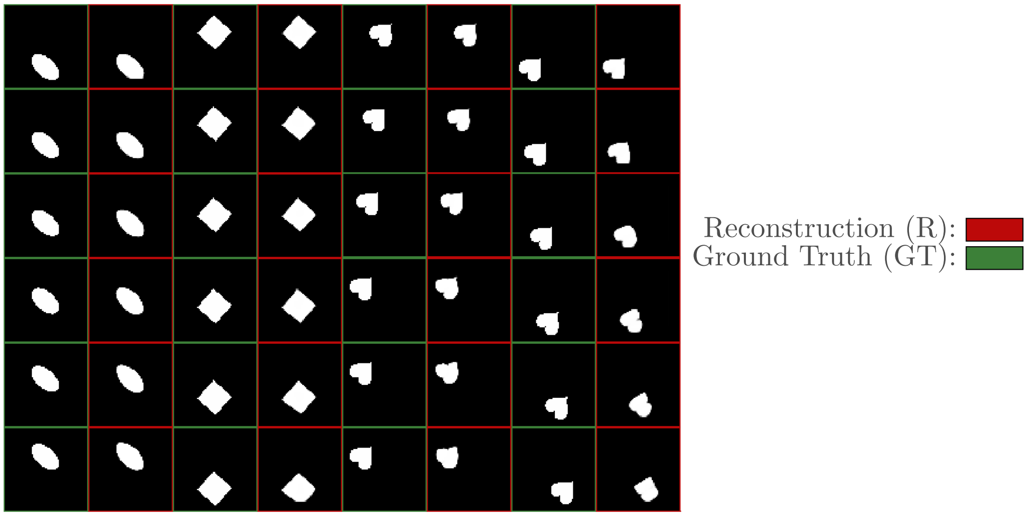


Figure 35: This figure illustrates the sequences of reconstructed images generated by the DAI after 500K training iterations. The columns alternate between the ground truth images and the reconstructed images. Time passes vertically (from top to bottom), and within each column, the same action is executed repeatedly.

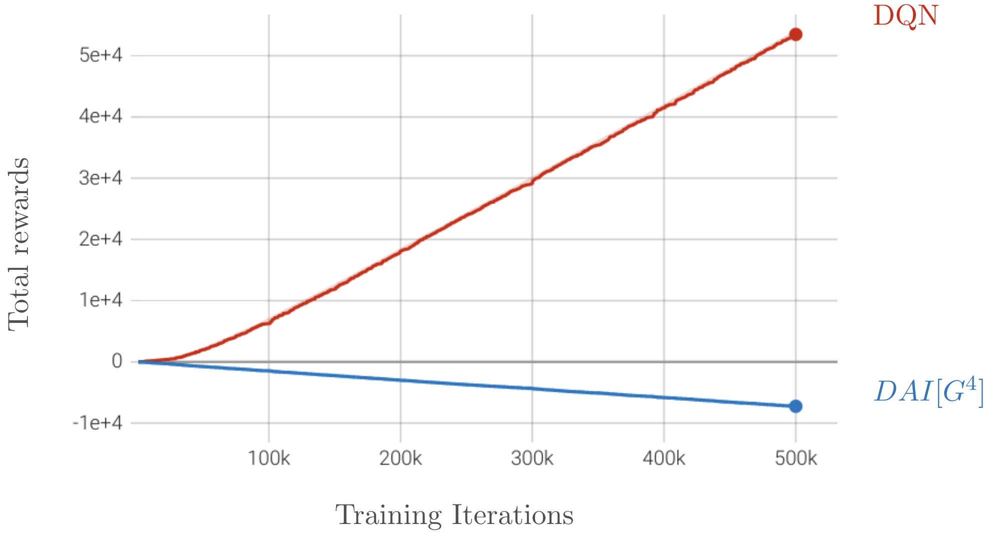


Figure 36: This figure illustrates the total amount of reward gathered by a DAI agent during the 500K iterations of training. This agent was maximising rewards while sampling actions from a softmax function of the policy network output. Put simply, the DAI agent does not solve the task and performs at the level of a random agent.

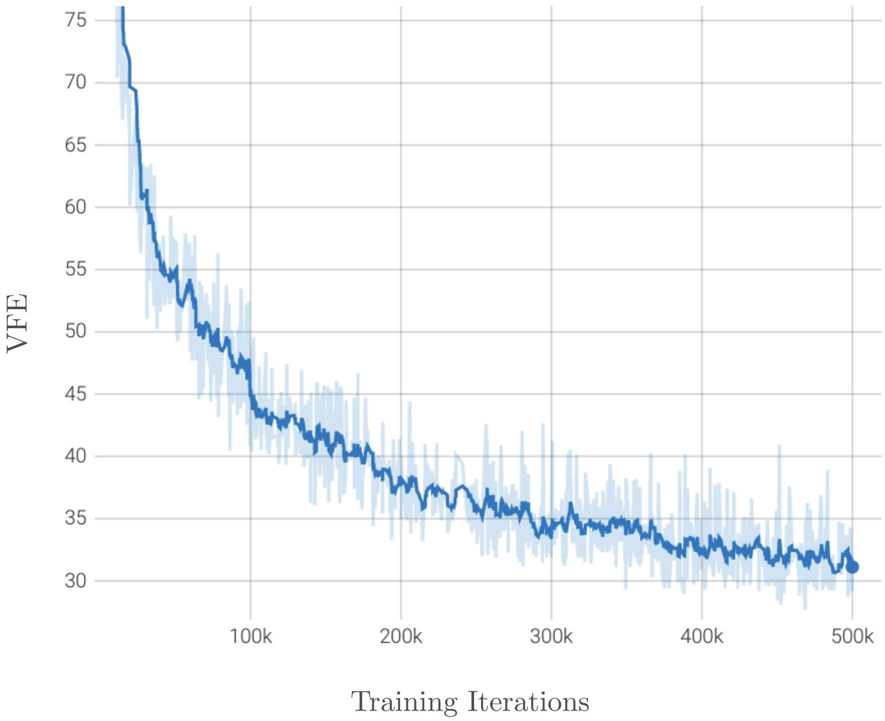
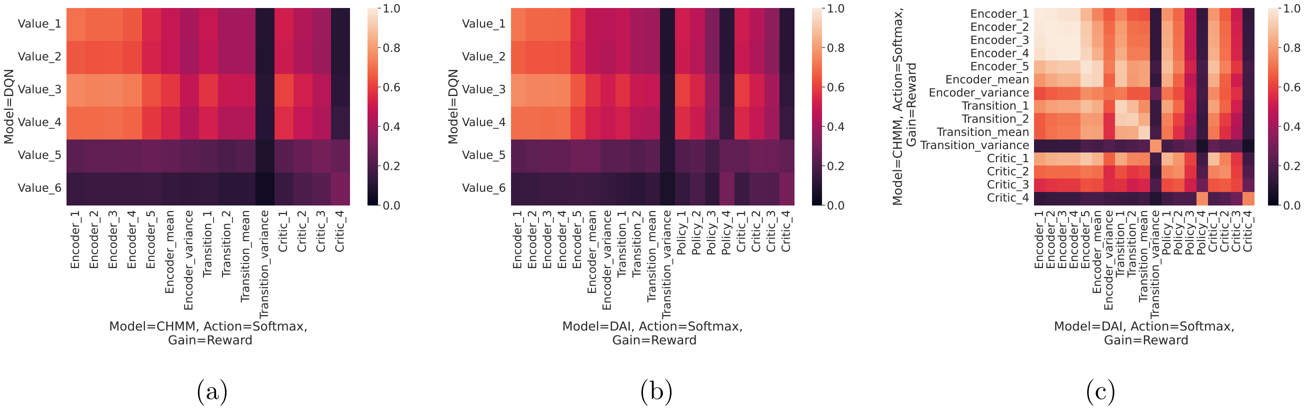


Figure 37: This figure illustrates the variational free energy of the DAI agent during the 500K iterations of training. This agent was maximising reward while sampling actions from a softmax function of the policy network output. The agent was able to minimise its variational free energy.



(a)

(b)

(c)

Figure 38: (a) shows the similarity between the representations learned by a DQN and a CHMM maximising the reward and using softmax action selection. (b) shows the similarity between the representations learned by a DQN and a DAI maximising the reward and using softmax action selection. (c) shows the similarity between the representations learned by a CHMM and a DAI. Both maximise the reward and use softmax action selection.

5. Conclusion

In this paper, we challenged the common assumption that deep active inference is a solved problem, by highlighting the challenges that need to be resolved for the field to move forward. We reviewed eight approaches implementing deep active inference: (a) DAI_{MC} by Fountas et al. (2020), (b) DAI_{VPG} by Millidge (2020), (c) DAI_{RHI} by Rood et al. (2020), (d) DAI_{HR} by Sancaktar et al. (2020), (e) DAI_{FA} by Ueltzhöffer (2018), (f) DAI_{POMDP} by van der Himst and Lanillos (2020), (g) DAI_{SSM} by Çatal et al. (2020), and (h) the approach by Schneider et al. (2022) for which the code was not available online.

Overall, those approaches brought interesting ideas such as: using deep neural networks to predict the parameters of the distributions of interest, using Monte-Carlo tree search for planning in active inference, and using a bootstrap estimate of the expected free energy to train a critic network. Yet, we struggled to replicate some of the results claimed, e.g. training DAI_{MC} on the animal AI environment, and getting access to the code of Schneider et al. (2022) was not possible. Ideally, future research should draw inspiration from the open science framework, by making the code that produced the claimed results open source. Also, the definition of the expected free energy varied between papers. This suggests that additional research is required to clarify the definition and justification of the expected free energy both in tabular and deep active inference. To sum up, recent research on deep active inference (Fountas et al., 2020; Millidge, 2020; Rood et al., 2020; Sancaktar et al., 2020; Ueltzhöffer, 2018; van der Himst and Lanillos, 2020; Çatal et al., 2020) has made an important first step towards a complete deep active inference agent, but a number of details still need to be honed, e.g., the definition and derivation of the expected free energy, the correctness of the implemention, and reproducibilty of research. For more details, the reader is referred to the subsections of Section 2 corresponding to the approaches just mentioned.

After reviewing existing research, we tried to progressively implement a deep active inference agent. First, we produced a variational auto-encoder agent that takes random actions. This agent was able to learn a useful (latent) representation of the task. However, since the agent takes random actions, it was unable to solve the task. Then, we added the transition network to create the hidden Markov model (HMM) agent, which also takes random actions. The agent was able to learn a good represenation of the task and of its dynamics, but was not able to solve the task.

Next, we tried to incorporate a critic network into the approach leading to the critical hidden Markov model (CHMM). In this context, we experimented with several possible implementations of the expected free energy. We also tried to remove the information gain and simply predict the reward. Additionally, we implemented three types of action selection strategies, namely: best action according to the expected free energy, softmax sampling, and epsilon-greedy.

When the epsilon-greedy algorithm was used, only the agent maximising reward was able to solve the task. However, the agent requires more training iterations than a simple deep Q-network to learn the right behaviour. This may be explained by the fact that the CHMM not only has to learn to solve the task, but also learn the dynamics of the environment. When softmax sampling was used, all the agents failed to solve the task. One of them perfoming even worse than an agent selecting random actions. Lastly, when selecting the best action, only the reward maximising agent was able to solve the task. Importantly, according to our experiments, the agent using the epsilon-greedy algorithm received the highest amount of cumulated rewards and learned to solve the task the fastest. Additionally, the reward maximising agents properly solve the task, but the quality of their latent representation is not as good as that of an HMM agent. This may be due to the fact that when performing reward maximising actions, the data available to learn the model of the environment lacks diversity, i.e., not enough exploration.

Next, we tried to incorporate a policy network into the approach leading to a complete deep active inference agent (DAI). As for the CHMM agent, we experiemented with several possible implementations of the expected free energy, tried to remove the information gain and simply predict the reward, and implemented three types of action selection strategies, namely: best action according to the expected free energy, softmax sampling, and epsilon-greedy algorithm. When the epsilon-greedy algorithm was used or the best action was selected, all agents failed to solve the task. When using softmax sampling, most of the agents were numerically unstable and crashed, and the remaining agents failed to solve the task.

Finally, we compared the similarity of the representation learned by the layers of various models (e.g., deep Q-network, CHMM, DAI, etc...) using centered kernel alignment. This reveals that the DQN learns general features in its first few layers, and very specialised features in its last two layers. The VAE learns similar features to the DQN in the first layers, but differs from the DQN in the last two layers; reflecting the difference in learning objectives. Similarly, the HMM learns similar features to the DQN in the first layers, but differs from the representation learned by the DQN in the last two layers. Also, the mean and variance representations learned by the HMM are different from their VAE counterparts, which suggests that the transition network influences the latent representation of the model.

Additionally, when using the $\mathring{\epsilon}$-greedy algorithm for action selection, the representations learned by the CHMM maximising reward is closer to the DQN than the CHMM minimising expected free energy is to the DQN. Importantly, the critic network of the reward maximising CHMM retains more information from the encoder than the CHMM minimising expected free energy. When the best action (according to the critic network) is selected, the CHMM maximising reward and the CHMM minimising expected free energy learn very similar representations except for the variance layers of the transition and encoder network. While performing further inspection of those (variance) layers, we found that the transition network of the reward maximising CHMM is a lot more certain than the transition network of the CHMM minimising expected free energy. More precisely, the CHMM minimising expected free energy is only confident about the world transition when performing action down. This suggests that the CHMM minimising expected free energy always pick the action down, and does not gather enough data for the other actions. Visualising the distribution of actions selected as training progresses corroborates this story by showing that the agent minimising EFE almost exclusively picks action down. In contrast, the CHMM maximising reward, keeps on selecting the actions left and right, which enables it to successfully solve the task. The only difference between those two CHMMs is the epistemic value, which aims to make the outputs of the transition and encoder network as close as possible. Thus, the CHMM minimising expected free energy is picking a single action (down), and becomes an expert at predicting the future when selecting this action. This effectively makes the KL divergence between the output of the transition and encoder networks small. Additionally, when selecting the action down, the average reward is zero, while for all the other actions, the expected reward will be negative. Therefore, if the CHMM has to stick to a single action to keep the KL divergence small, then the action down is the most rewarding.

The same observations about the variance layers also applies to CHMMs using softmax sampling for action selection. While this may explain why the model optimising the expected free energy does not solve the task, it does not explain why the model maximising the reward cannot solve the task, and we can hypothesise that those results may be due to the softmax action selection. More precisely, if the values predicted by the critic network are very close to each other, then an agent using softmax sampling may perform random actions. Note, increasing the gain parameter may help the agent to differentiate between values close to each other.

Lastly, the representational similarity between the DAI (maximising reward using softmax sampling) and DQN is very close to the representational similarity between a DQN with a CHMM (maximising reward using softmax sampling). Also, the DAI's policy and critic network learn similar representations, which indicates that the policy network is learning correctly. Thus, the fact that the DAI (maximising reward using softmax sampling) fails in the dSprites environment is likely due to the softmax action selection and not to the representation learned by the model.

To conclude, the field of deep active inference has benefited from a large variety of ideas from the reinforcement and deep learning literature. In the future, it would be valuable to provide an approach that satisfies the following five desirata: (i) the approach is complete, i.e., it is composed of an encoder, a decoder, a transition network, a policy network and (optionally) a critic network, (ii) the mathematics underlying the approach is ~~principled and errorless~~ errorless and consistent with the free energy principle, (iii) the implementation ~~is correct, and~~ consistent with the mathematics, (iv) the code is publicly available so that the correctness of the implementation can be verified and the results reproduced, and (v) the approach is able to solve tasks with a large input space, e.g. image-based tasks. We believe that such an approach will benefit the field of deep active inference by providing a strong and reproducible baseline against which future research could benchmark.

Acknowledgments

TO BE FILLED

References

Richard Bellman. On the theory of dynamic programming. *Proc Natl Acad Sci U S A*, 38(8):716–719, August 1952.

Lisa Bonheme and Marek Grzes. How do Variational Autoencoders Learn? Insights from Representational Similarity. *arXiv e-prints*, art. arXiv:2205.08399, May 2022.

C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

Ozan Çatal, Samuel Wauthier, Cedric De Boom, Tim Verbelen, and Bart Dhoedt. Learning generative state space models for active inference. *Frontiers in Computational Neuroscience*, 14:574372, 2020.

Théophile Champion, Howard Bowman, and Marek Grześ. Branching time active inference: empirical study, 2021a.

Théophile Champion, Howard Bowman, and Marek Grześ. Branching time active inference: the theory and its generality, 2021b.

Théophile Champion, Marek Grześ, and Howard Bowman. Realising active inference in variational message passing: the outcome-blind certainty seeker, 2021c.

Théophile Champion, Marek Grześ, and Howard Bowman. Branching Time Active Inference with Bayesian Filtering, 2021d.

Théophile Champion, Marek Grześ, and Howard Bowman. Multi-modal and multi-factor branching time active inference, 2022. URL `https://arxiv.org/abs/2206.12503`.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for Learning Kernels Based on Centered Alignment. *J. Mach. Learn. Res.*, 13(1):795–828, March 2012. ISSN 1532-4435.

Lancelot Da Costa, Thomas Parr, Noor Sajid, Sebastijan Veselic, Victorita Neacsu, and Karl Friston. Active inference on discrete state-spaces: a synthesis, 2020a.

Lancelot Da Costa, Noor Sajid, Thomas Parr, Karl Friston, and Ryan Smith. The relationship between dynamic programming and active inference: the discrete, finite-horizon case, 2020b.

Nello Cristianini, John Shawe-Taylor, André Elisseeff, and Jaz S Kandola. On Kernel-Target Alignment. In T G Dietterich, S Becker, and Z Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14, pages 367–373. MIT Press, Vancouver, Canada, 2002. URL http://papers.nips.cc/paper/1946-on-kernel-target-alignment.pdf.

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. ISSN 2451-9022. doi: https://doi.org/10.1016/j.bpsc.2018.06.010. URL http://www.sciencedirect.com/science/article/pii/S2451902218301617. Computational Methods and Modeling in Psychiatry.

Carl Doersch. Tutorial on variational autoencoders, 2016.

Thomas H. B. FitzGerald, Raymond J. Dolan, and Karl Friston. Dopamine, reward learning, and active inference. *Frontiers in Computational Neuroscience*, 9:136, 2015. ISSN 1662-5188. doi: 10.3389/fncom.2015.00136. URL https://www.frontiersin.org/article/10.3389/fncom.2015.00136.

Zafeirios Fountas, Noor Sajid, Pedro A. M. Mediano, and Karl Friston. Deep active inference agents using Monte-Carlo methods, 2020.

Karl Friston, Thomas FitzGerald, Francesco Rigoli, Philipp Schwartenbeck, John O Doherty, and Giovanni Pezzulo. Active inference and learning. *Neuroscience & Biobehavioral Reviews*, 68:862 – 879, 2016. ISSN 0149-7634. doi: https://doi.org/10.1016/j.neubiorev.2016.06.022.

Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring Statistical Dependence with Hilbert-Schmidt Norms. In Sanjay Jain, Hans Ulrich Simon, and Etsuji Tomita, editors, *Algorithmic Learning Theory*, pages 63–77. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-31696-1.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Sy2fzU9gl.

Laurent Itti and Pierre Baldi. Bayesian surprise attracts human attention. *Vision Research*, 49(10):1295 – 1306, 2009. ISSN 0042-6989. doi: https://doi.org/10.1016/j.visres.2008.09.007. URL http://www.sciencedirect.com/science/article/pii/S0042698908004380. Visual Attention: Psychophysics, electrophysiology and neuroimaging.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, volume 2, Banff, Canada, 2014. URL http://arxiv.org/abs/1312.6114.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques.* MIT press, 2009.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of Neural Network Representations Revisited. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3519–3529, Long Beach, USA, Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/kornblith19a.html.

Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning, 2016.

Pablo Lanillos, Gordon Cheng, et al. Robot self/other distinction: active inference meets neural networks learning in a mirror. *arXiv preprint arXiv:2004.05473*, 2020.

Niru Maheswaranathan, Alex Williams, Matthew Golub, Surya Ganguli, and David Sussillo. Universality and individuality in neural dynamics across large populations of recurrent networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing*

Systems, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/5f5d472067f77b5c88f69f1bcfda1e08-Paper.pdf.

Loic Matthey, Irina Higgins, Demis Hassabis, and Alexander Lerchner. dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/, 2017.

Beren Millidge. Combining active inference and hierarchical predictive coding: A tutorial introduction and case study., 2019. URL https://doi.org/10.31234/osf.io/kf6wc.

Beren Millidge. Deep active inference as variational policy gradients. *Journal of Mathematical Psychology*, 96:102348, 2020. ISSN 0022-2496. doi: https://doi.org/10.1016/j.jmp.2020.102348. URL http://www.sciencedirect.com/science/article/pii/S0022249620300298.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

Guillermo Oliver, Pablo Lanillos, and Gordon Cheng. Active inference body perception and action for humanoid robots. *arXiv preprint arXiv:1906.03022*, 2019.

Thomas Parr and Karl J Friston. Generalised free energy and active inference: can the future cause the past? *bioRxiv*, 2018. doi: 10.1101/304782. URL https://www.biorxiv.org/content/early/2018/04/23/304782.

Corrado Pezzato, Carlos Hernandez, and Martijn Wisse. Active inference and behavior trees for reactive action planning and execution in robotics, 2020.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In Eric P Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China, 2014. PMLR. URL http://proceedings.mlr.press/v32/rezende14.html.

P. Robert and Y. Escoufier. A Unifying Tool for Linear Multivariate Statistical Methods: The RV- Coefficient. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 25(3):257–265, 1976. ISSN 00359254, 14679876. URL http://www.jstor.org/stable/2347233.

Thomas Rood, Marcel van Gerven, and Pablo Lanillos. A deep active inference model of the rubber-hand illusion. In Tim Verbelen, Pablo Lanillos, Christopher L. Buckley, and Cedric De Boom, editors, *Active Inference*, pages 84–91, Cham, 2020. Springer International Publishing. ISBN 978-3-030-64919-7.

Cansu Sancaktar, Marcel van Gerven, and Pablo Lanillos. End-to-end pixel-based deep active inference for body perception and action, 2020.

Tim Schneider, Boris Belousov, Hany Abdulsamad, and Jan Peters. Active inference for robotic manipulation. *arXiv preprint arXiv:2206.10313*, 2022.

Philipp Schwartenbeck, Johannes Passecker, Tobias U Hauser, Thomas H B FitzGerald, Martin Kronbichler, and Karl Friston. Computational mechanisms of curiosity and goal-directed exploration. *bioRxiv*, 2018. doi: 10.1101/411272. URL https://www.biorxiv.org/content/early/2018/09/07/411272.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587): 484–489, 2016. doi: 10.1038/nature16961. URL https://doi.org/10.1038/nature16961.

Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*. MIT press Cambridge, 1998.

Kai Ueltzhöffer. Deep active inference. *Biol. Cybern.*, 112(6):547–573, dec 2018. ISSN 0340-1200. doi: 10.1007/s00422-018-0785-7. URL `https://doi.org/10.1007/s00422-018-0785-7`.

Otto van der Himst and Pablo Lanillos. Deep active inference for partially observable mdps. *CoRR*, abs/2009.03622, 2020. URL `https://arxiv.org/abs/2009.03622`.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning, 2015.

Appendix A: Notation

Symbol	Meaning
s_τ, o_τ, r_τ, a_τ	State, observation, reward and action at time step τ, respectively.
o^r_τ, $\hat{s}^r_\tau$, $\mathring{o}^r_\tau$	An observation in which a reward has been encoded as explained in Figure 40, the hidden state sampled from the encoder when feeding o^r_τ as input, and the observation reconstructed by the decoder from the output of the transition network, respectively.
$\hat{s}_\tau$, $\hat{o}_\tau$	A state sampled from the encoder at time step τ when o_τ is provided as input, and the image reconstructed by the decoder at time τ when using $\hat{s}_\tau$ as input, respectively.
$\mathring{o}_\tau$	The observations at time step τ predicted by the transition network.
$s_{i:j}$, $o_{i:j}$, $a_{i:j}$	Respectively, the set of states, observations, and actions between time step i and j (included).
π, π_τ, π'	A policy, i.e. a sequence of actions, the action prescribed by the policy at time step τ, and another policy whose size is smaller or equal than the size of π, respectively.
$\mathcal{A}$, Π	The set of possible actions, and the set of possible policies, respectively.
$\mathbb{A}_s$	The set of all anscestors of a node s.
$\#A, \#C, \#S, \#O$	The number of actions, channels, states and observations, respectively.
$\mathcal{Q}_{\theta_a}$, $\hat{\mathcal{Q}}_{\hat{\theta}_a}$	The Q-network parameterised by θ_a, and the target network parameterised by $\hat{\theta}_a$
$\mathcal{G}_{\theta_a}$, $\hat{\mathcal{G}}_{\hat{\theta}_a}$	The critic network parameterised by θ_a, and the target network parameterised by $\hat{\theta}_a$.
$\mathcal{E}_{\phi_s}$, $\mathcal{D}_{\theta_o}$	The encoder network parameterised by ϕ_s, and the decoder network parameterised by θ_o.
$\mathcal{P}_{\phi_a}$	The policy network parameterised by ϕ_a.
$\mathcal{T}_{\theta_s}$, $\mathcal{T}_{\theta_o}$	The transition network parameterised by θ_s or θ_o, respectively.
θ, ϕ	All the parameters of the generative model, and the variational distribution, respectively.
a, b, c, d	Four hyperparameters involved in the computation of ω_t.
$N(s_\tau, a_\tau)$	The number of times action a_τ was explored in state s_τ.
t, γ	The present time step, and the discount factor, respectively.
ζ, ψ	The precision of the prior over actions, and the precision of the prior preferences.
ϵ, $\hat{\epsilon}$	The random variable used in the re-parameterisation trick, and a sample of epsilon.
$\mathring{\epsilon}$	The probability of selecting a random actions when using the $\mathring{\epsilon}$-greedy algorithm.
T_{dec}	A hyperparameter defining the threshold value corresponding to a clear winner during MCTS.
$G(\pi)$, $G_\tau(\pi)$	The expected free energy (EFE) of policy π, and the EFE received at time step τ when following policy π, respectively.
$\bar{\boldsymbol{G}}_s$, $\boldsymbol{G}_s^{\text{aggr}}$, $\boldsymbol{G}_s$, $\boldsymbol{N}_s$	The average EFE, the aggregated EFE, the EFE, and the number of visits of a node s, respectively.
$\mu_o, \sigma_o, \mu_a, \sigma_a$	The mean and variance vectors predicted by the encoder and policy networks of the DAI_{FA}.
μ, σ	The mean and variance of the Gaussian distribution over s_t predicted by the encoder.
$\mathring{\mu}$, $\mathring{\sigma}$	The mean and variance of the Gaussian distribution over s_{t+1} predicted by the transition.
$\hat{\mu}$, $\hat{\sigma}$	The mean and variance of the Gaussian distribution over s_{t+1} predicted by the encoder.
$\hat{\pi}$	The parameters of the categorical distribution over a_t predicted by the policy network.
ω_t	The top-down attention parameter modulating the precision of the transition mapping.
$[\,\text{condition}\,]$	An indicator function that equals one if the condition is satisfied and zero otherwise.
$\sigma[\bullet]$	The softmax function.
$\mathrm{Cat}(x; \phi_x)$	A categorical distribution over x parameterised by ϕ_x.
$\mathrm{Bernoulli}(x; \phi_x)$	A Bernoulli distribution over x parameterised by ϕ_x.
$\mathcal{B}\mathrm{ernoulli}(x; \phi_x)$	A product of Bernoulli distributions over x parameterised by ϕ_x.
$\mathcal{N}(x; \mu_x, \sigma_x)$	A multivariate Gaussian over x parameterised by a mean vector μ_x, and a diagonal covariance matrix whose diagonal elements are σ_x.
$X \overset{i}{\to} Y$, $X \overset{m}{\to} Y$	X is fed as *input* to Y, and the *mean* of the distribution predicted by X is fed as input to Y.
$X \overset{s}{\to} Y$, $X \to Y$	a *sample* from the distribution predicted by X is fed as input to Y, and X outputs Y.

Table 1: Notation of Sections 2 and 3.

Appendix B: DAI_{MC} discrepancies between the paper and the code

In this section, we focus on the authors' implementation of DAI_{MC} available on GitHub: `https://github.com/zfountas/deep-active-inference-mc/`. First, according to a private communication with one of the authors, the code available on GitHub (on the 6th of June 2022) is not the same as the one used to run the experiments of the paper. Below, we describe the discrepancies between the paper and the code. For example, the computation of ω_t in the paper is as follows:

$$\omega_t = \frac{a}{1 + \exp(-\frac{b - D_t}{c})} + d,$$

while the code uses the following formula:

$$\omega_t = a \times \left(1 - \frac{1}{1 + \exp\big(-\frac{D_t - b}{c}\big)}\right) + d.$$

Also, the paper states that MCTS is perfomed to compute the prior over policies during training. However, in the code, MCTS is only used when testing the model, i.e., no MCTS when training the agent. Additionally, the paper states that actions are selected by sampling from:

$$\tilde{P}(a_t) = \frac{N(\hat{s}_t, a_t)}{\sum_{\hat{a}_t} N(\hat{s}_t, \hat{a}_t)}.$$

However, the code selects an entire sequence of actions $\pi = (\mathring{a}_t, \mathring{a}_{t+1}, ..., \mathring{a}_{t+n})$ recursively from the root node in the tree. At each step in the recursion, the node with the highest number of visits $\mathring{a}_\tau$ is selected. Then, actions cancelling each other are removed from the sequence, e.g., if $a_\tau = LEFT$ and $a_{\tau+1} = RIGHT$ then both actions are removed from the sequence. This procedure generates a new sequence of actions π' of equal or smaller length. Finally, the entire sequence of actions π' is performed in the environment. This avoids the repetition of the planning process for each action-perception cycle (saving computational time), however, this also requires domain knowledge (to remove actions that cancel each other out).

Additionally, in the paper, experiments are run ~~in~~ on both the dSprites environment and the animal AI environment. However, the code does not allow the replication of the results on the animal AI environment, i.e., the code handling the animal AI environment has been removed. In addition, the evaluation of the expected free energy is non trivial (see below) and the details are not discussed in the paper. Before explaining how the terms of the EFE are computed, we introduce notation that allows us to express those computational steps concisely. For example, we note:

$$o^r_t \overset{i}{\to} \text{Encoder} \overset{s}{\to} \text{Transition} \overset{m}{\to} \mathring{s}^r_{t+1},$$

meaning that o^r_t is used as *input* ($\overset{i}{\to}$) for the encoder, then a state is *sampled* ($\overset{s}{\to}$) from the distribution predicted by the encoder and used as input for the transition network, finally, the *mean* ($\overset{m}{\to}$) of the distribution predicted by the transition network is used as a maximum aposteriori estimate of $\mathring{s}^r_{t+1}$. Note, the transition network takes two inputs (i.e., a state and an action), when using our concise notation we implicitly assume that the actions prescribed by the policy[8] π are provided as input to the transition network. Also, for each time step τ, the reward r_τ collected by the agent is encoded in the pixels of the image o_τ as explained in Figure 40, leading to a new image o^r_τ. As illustrated on the right of Figure 39, the encoder/decoder networks are trained to predict the resulting images o^r_τ. The computation of the first term in equation (8) is illustrated on the left of Figure 39. Concisely, we have:

$$o^r_t \overset{i}{\to} \text{Encoder} \overset{s}{\to} \text{Transition} \overset{s}{\to} \text{Decoder} \overset{m}{\to} \mathring{o}^r_{t+1}.$$

Next, a matrix ($\mathring{r}_{t+1}$) encoding the maximum reward that the agent can gather is used as parameter of Bernoulli distributions to compute the logarithm of the probability (i.e., $\boldsymbol{L}$) of the three first rows of the reconstructed image $\mathring{o}^r_{t+1}$. Note, as explained in Figure 40, the first three rows contain the predicted reward obtained at time $t + 1$.

[8] π is the policy for which the expected free energy is being computed.

Finally, the mean of $\boldsymbol{L}$ is then computed and is multiplied by ten to get $\mathbb{E}_{\tilde{Q}}[\ln \tilde{P}(o_\tau|\pi)]$. Similarly, the computation of $H[Q(s_\tau|\pi)]$ proceeds as follows:

$$o_t^r \xrightarrow{i} \text{Encoder} \xrightarrow{s} \text{Transition} \rightarrow \mathring{\mu}, \ln \mathring{\sigma},$$

where $Q(s_\tau|\pi)$ is equated with $\mathcal{N}(s_\tau; \mathring{\mu}, \mathring{\sigma})$, and an analytical solution is used to compute the entropy of $Q(s_\tau|\pi)$. Next, the computation of $H[Q(o_\tau|s_\tau, \pi)]$ goes as follows:

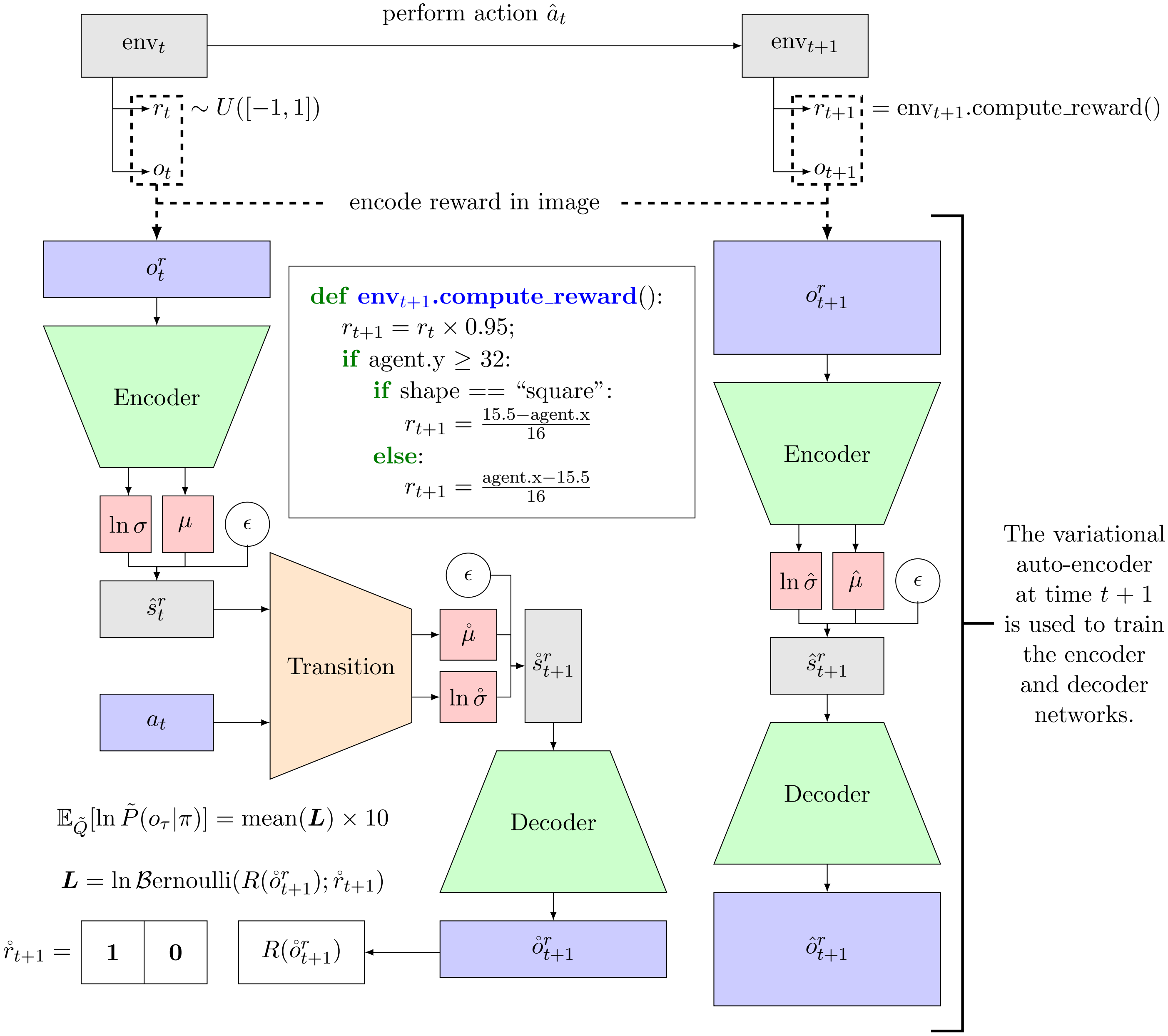
$$o_t^r \xrightarrow{i} \text{Encoder} \xrightarrow{s} \text{Transition} \xrightarrow{s} \text{Decoder} \xrightarrow{m} \mathring{o}_{t+1}^r,$$

where observation $\mathring{o}_{t+1}^r$ is equated to the parameters of the Bernoulli distribution $Q(o_\tau|s_\tau, \pi)$, and an analytical solution is used to compute $H[Q(o_\tau|s_\tau, \pi)]$. Surprisingly, another observation $\mathring{o}_{t+1}^r$ sampled exactly as before is equated to the parameters of the Bernoulli distribution $Q(o_\tau|s_\tau, \theta, \pi)$, and the same analytical solution is used to compute $H[Q(o_\tau|s_\tau, \theta, \pi)]$. Finally, $H[Q(s_\tau|o_\tau, \pi)]$ is computed by feeding $\mathring{o}_{t+1}^r$ back into the encoder to get the mean and log-variance of the Gaussian distribution $Q(s_\tau|o_\tau, \pi)$, and the analytical solution for the entropy of a Gaussian is used to compute $H[Q(s_\tau|o_\tau, \pi)]$.

In summary, two samples of $\mathring{o}_{t+1}^r$ (sampled as described in Figure 40) have been equated to the parameters of two diferent distributions, i.e., $Q(o_\tau|s_\tau, \theta, \pi)$, and $Q(o_\tau|s_\tau, \pi)$. Additionally, a third sample of $\mathring{o}_{t+1}^r$ (sampled in the same way) has also been used as input to the distribution $\tilde{P}(o_\tau|\pi)$. Lastly, while Fountas et al. (2020) defines the EFE as in (8), the code turns a plus into a minus, leading to the following definition of the EFE:

$$\begin{aligned} G_\tau(\pi) = &- \mathbb{E}_{\tilde{Q}}\Big[\ln \tilde{P}(o_\tau|\pi)\Big] \\ &- \mathbb{E}_{Q(\theta|\pi)}\Big[\mathbb{E}_{Q(o_\tau|\theta,\pi)}\big[H[Q(s_\tau|o_\tau, \pi)]\big] - H[Q(s_\tau|\pi)]\Big] \\ &+ \mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)}\Big[H[Q(o_\tau|s_\tau, \theta, \pi)]\Big] - \mathbb{E}_{Q(s_\tau|\pi)}\Big[H\big[Q(o_\tau|s_\tau, \pi)\big]\Big], \end{aligned}$$

where the red minus should be a plus.


perform action $\hat{a}_t$
env_t
env_{t+1}
$r_t \sim U([-1,1])$
o_t
$r_{t+1} = \text{env}_{t+1}.\text{compute\_reward}()$
o_{t+1}
encode reward in image
o_t^r
Encoder
$\ln\sigma$
μ
ϵ
$\hat{s}_t^r$
a_t
Transition
ϵ
$\mathring{\mu}$
$\ln\mathring{\sigma}$
$\mathring{s}_{t+1}^r$
Decoder
$\mathring{o}_{t+1}^r$
$R(\mathring{o}_{t+1}^r)$
$\mathring{r}_{t+1} =$
1
0
$\mathbb{E}_{\tilde{Q}}[\ln\tilde{P}(o_\tau|\pi)] = \text{mean}(\boldsymbol{L}) \times 10$
$\boldsymbol{L} = \ln\mathcal{B}\text{ernoulli}(R(\mathring{o}_{t+1}^r); \mathring{r}_{t+1})$
o_{t+1}^r
Encoder
$\ln\hat{\sigma}$
$\hat{\mu}$
ϵ
$\hat{s}_{t+1}^r$
Decoder
$\hat{o}_{t+1}^r$
The variational auto-encoder at time $t+1$ is used to train the encoder and decoder networks.

```
def env_{t+1}.compute_reward():
    $r_{t+1} = r_t \times 0.95$;
    if agent.y $\geq 32$:
        if shape == "square":
            $r_{t+1} = \frac{15.5-\text{agent.x}}{16}$
        else:
            $r_{t+1} = \frac{\text{agent.x}-15.5}{16}$
```

Figure 39: This figure illustrates the computation of $\mathbb{E}_{\tilde{Q}}[\ln\tilde{P}(o_\tau|\pi)]$ in (the code of) Fountas et al. (2020). The environment at time t provides the agent with an image o_t and a reward r_t randomly sampled from the interval $[-1;1]$. Then, action $\hat{a}_t$ is performed in the environment and the agent observes an image o_{t+1} and a reward r_{t+1}, where r_{t+1} is computed according to the function presented in the center of the image. Next, the reward at time t and $t+1$ are encoded in the images received at time t and $t+1$, respectively, c.f., Figure 40 for details about the encoding. The encoded image at time $t+1$ (i.e., o_{t+1}^r) is then fed into the encoder, the re-parameterisation trick is then used to sample a state from the variational posterior. This states is fed into the decoder which tries to reconstruct the image inputed into the encoder. Once $\hat{o}_{t+1}^r$ has been computed, the weights of the encoder and decoder are learned using back-propagation. On the other hand, the encoded image at time t (i.e., o_t^r) is used to compute $\mathbb{E}_{\tilde{Q}}[\ln\tilde{P}(o_\tau|\pi)]$. More precisely, the o_t^r is fed into the encoder, and a state is sampled from the variational posterior $Q_{\phi_s}(s_t)$. This state is then fed as input into the transition network along with the action prescribed by π at time t, i.e., a_t. A state at time $t+1$ can then be sampled from the distribution predicted by the transition network. This state is then inputed into the decoder, which outputs $\mathring{o}_{t+1}^r$. Next, a matrix (i.e., $\mathring{r}_{t+1}$) encoding the maximum reward that the agent can gather is used as a parameter of a Bernoulli distribution to compute the logarithm of the probability (i.e., $\boldsymbol{L}$) of the first three rows of $\mathring{o}_{t+1}^r$, i.e., $R(\mathring{o}_{t+1}^r)$. The mean of $\boldsymbol{L}$ is then computed and is multiplied by ten to get $\mathbb{E}_{\tilde{Q}}[\ln\tilde{P}(o_\tau|\pi)]$.

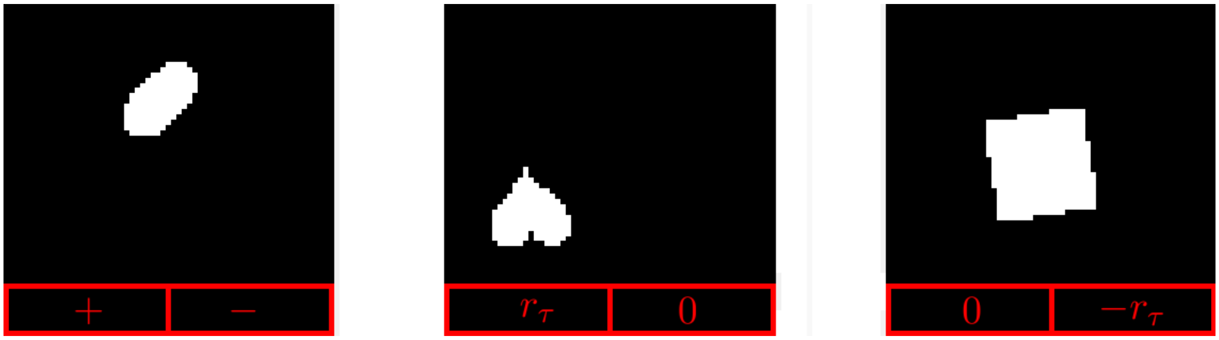


Figure 40: This figure illustrates how the reward $r_\tau \in [-1, 1]$ is encoded in image o_τ. On the left, the plus and minus signs shows where the reward will be encoded in the image if the reward is positive or negative, respectively. In the middle, a positive reward is being encoded on the left side of the image. On the right, a negative reward is being encoded on the right of the image.