Deconstructing deep active inference:

a contrarian information gatherer.

Théophile Champion TXC314@STUDENT.BHAM.AC.UK

University of Birmingham, School of Computer Science

Birmingham B15 2TT, 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@BHAM.AC.UK

University of Birmingham, School of Psychology and Computer Science,

Birmingham B15 2TT, United Kingdom

University College London, Wellcome Centre for Human Neuroimaging (honorary)

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 research 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 progressively build a deep active inference agent as follows: we (i) implement a variational auto-encoder (VAE), (ii) implement a deep hidden Markov model (HMM), and (iii) implement a deep critical hidden Markov model (CHMM). For the CHMM, we implemented two versions, one minimizing expected free energy, i.e., CHMM[EFE], and one maximizing rewards, i.e., CHMM[reward], then we experimented with three different action selection strategies, i.e., the ϵ-greedy algorithm as well as softmax and best action selection. According to our experiments, the models able to solve the dSprites environment are the ones that maximize rewards. On further inspection, we found that the CHMM minimizing expected free energy almost always picks the same action, which makes it unable to solve the dSprites environment. In contrast, the CHMM maximizing reward, keeps on selecting all the actions, enabling 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 networks as close as possible. Thus, the CHMM minimizing expected free energy repeatedly picks a single action, 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. We also show in simulation that the epistemic value used in deep active inference can behave degenerately and in certain circumstances, effectively lose, rather than gain, information. As the agent minimizing EFE is not able to explore its environment, 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, 2020; Champion et al, 2021, 2022b,a,c). 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; Kingma and Welling, 2014; Rezende et al, 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 (Browne et al, 2012; Silver et al, 2016) has been used to improve planning efficency (Fountas et al, 2020; Champion et al, 2022b,a,c,d).

Another closely related field is reinforcement learning (Mnih et al, 2013; van Hasselt et al, 2015; Lample and Chaplot, 2016), where an agent must interact with its environment to maximize the amount of rewards it receives. At each time step τ, the agent is making an observation o_τ, and is allowed to perform one action a_τ. After performing a_τ in state o_τ, the agent receives a reward r_τ. The Q-learning algorithm (Sutton et al, 1998) aims to maximize reward by computing the Q-values $q(o_\tau, a_\tau)$, for each state-action pair $(o_\tau,\, a_\tau)$. 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 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 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.

A known challenge of reinforcement learning 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:

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

where t is the present time step, $\mathcal{A}$ is the set of available actions, $q(o_t, a_t)$ 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}.

Unfortunately, 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.,

$$q(o_t, a_t) = \mathbb{E}_{o_{t+1} \sim E(o_t, a_t)} \Big[r_t + \gamma \max_{a_{t+1} \in \mathcal{A}} \hat{\mathcal{Q}}_{\hat{\theta}_a}(o_{t+1}, a_{t+1}) \Big]. \tag{2}$$

[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 (Friston et al, 2016; Fountas et al, 2020; Millidge, 2020).

Two other reinforcement learning approaches were used as benchmarks, i.e., Proximal Policy Optimization (PPO) (Schulman et al, 2017), and a synchronous version of Asynchronous Advantage Actor Critic (A3C) (Mnih et al, 2016). A3C is an actor-critic approach, where the actor is trained to maximize a policy gradient estimator, i.e., $\mathbb{E}[\nabla_\theta \ln \pi_\theta(a_t|s_t)\hat{\boldsymbol{A}}]$, where $\pi_\theta(a_t|s_t)$ is a stochastic policy and $\hat{\boldsymbol{A}}$ is an estimator of the advantage function, where the advantage of an action is the difference between the Q-value of an action-state pair and the value of that state. Additionally, the critic network is used to estimate the advantage function. PPO works similarly, but improves upon A3C by optimizing a clipped surrogate objective.

In Section 2, we review the existing literature and present: 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. Further details of these approaches is presented in Champion et al (2023). We argue that while all these approaches illuminate important issues associated with realizing a deep active inference agent, a fully complete implementation[2] 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). Lastly, a critic network is added to define a probability distribution over actions, which leads to the critical HMM (CHMM). Importantly, building a VAE and a HMM first, we can validate

[2]A complete implementation of deep active inference would contain at least an encoder, a decoder, a transition and a critic network. It would also need to be compliant with the theory underlying active inference.

the implementation of individual components (i.e., encoder, decoder, and transition network), before tackling the CHMM, which represents a complete implementation of active inference. This method also allows us to better localize the source of potential problems. Then, in Section 4, we discuss our findings regarding the abilities and limitations of each intermediate step. Finally, in Section 5, we provide an analysis of the epistemic value, before putting our finding in context and concluding this paper, c.f., Section 6.

2. Review of existing research

In this section, we discuss 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 number of other deep active inference agents for which the code is unavailable. Note, figures illustrating these approaches can be found in Champion et al (2023). Importantly, in active inference, observations $o_\tau \in \mathbb{R}^{\#O}$ are assumed to be generated by latent states $s_\tau \in \mathbb{R}^{\#S}$, and states $s_{\tau+1}$ depends on the previous states s_τ and actions $a_\tau \in \mathbb{R}^{\#A}$. However, in most cases, $\#A = 1$ and actions are discrete, i.e., $a_\tau \in \mathbb{N}$.

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

In this section, we review the DAI_{MC} agent proposed by Fountas et al (2020), which represents the most ambitious and complete implementation of a deep active inference agent that accordingly adds important new concepts to the field. 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. The encoder $\mathcal{E}_{\phi_s}$ takes images as input, and outputs the mean and variance of the (Gaussian) variational distribution over hidden states, i.e., $Q_{\phi_s}(s_t) = \mathcal{N}(s_t; \mu, I\sigma)$, where $\mu, \sigma = \mathcal{E}_{\phi_s}(o_t)$ and $Q_{\phi_s}(s_t)$ denote the variational posterior. 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.,

$$P_{\theta_o}(o_t|s_t) = \mathcal{B}\text{ernoulli}(o_t; \hat{o}_t), \tag{3}$$

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:

$$\mathcal{B}\text{ernoulli}(o_t; \hat{o}_t) = \prod_{x,y} \text{Bernoulli}(o_t[x,y]; \hat{o}_t[x,y]), \tag{4}$$

where Bernoulli$(\bullet; \bullet)$ is a Bernoulli distribution over the possible values of the pixel $o_t[x,y]$, parameterized by $\hat{o}_t[x,y]$. 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., $P_{\theta_s}(s_{\tau+1}|s_\tau, a_\tau) = \mathcal{N}(s_{\tau+1}; \mathring{\mu}, I\frac{\mathring{\sigma}}{\omega_t})$, 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 categorical 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:

$$P(a_t) = \sum_{\pi \in \Pi} [\pi_t = a_t] P(\pi), \tag{5}$$

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:

$$P(\pi) = \sigma[-G(\pi)], \tag{6}$$

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

$$G(\pi) = \sum_{\tau=t}^{T} G_\tau(\pi) = \sum_{\tau=t}^{T} \mathbb{E}_{\tilde{Q}}\Big[\ln Q(s_\tau, \theta|\pi) - \ln \tilde{P}(o_\tau, s_\tau, \theta|\pi)\Big], \tag{7}$$

where $\theta = \{\theta_o, \theta_s\}$ are the generative model parameters, $\tilde{Q} = Q(o_\tau, s_\tau, \theta|\pi) = Q(\theta|\pi)Q(s_\tau|\theta, \pi)Q(o_\tau|s_\tau, \theta, \pi)$ is the predictive posterior, and $\tilde{P}(o_\tau, s_\tau, \theta|\pi) = P(\theta|s_\tau, o_\tau, \pi)P(s_\tau|o_\tau, \pi)P(o_\tau|\pi)$ is the target distribution. Finally, the top-down attention parameter is computed as follows:

$$\omega_t = \frac{a}{1 + \exp(-\frac{b-D_t}{c})} + d, \tag{8}$$

where $D_t = D_{\mathrm{KL}}\left[\, Q_{\phi_a}(a_t|s_t) || \, P(a_t)\right]$, and $\{a, b, c, d\}$ are fixed hyperparameters. Importantly, the above expression for ω_t is part of the model specification, i.e., it is not obtained through variational inference. 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 the paragraph before Equation 5) 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 5) 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 estimate the expected free energy of each action (see below). Finally, action selection is performed by sampling from the following distribution:

$$\tilde{P}(a_t) = \frac{N(\hat{s}_t, a_t)}{\sum_{\hat{a}_t} N(\hat{s}_t, \hat{a}_t)}, \tag{9}$$

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. Importantly, while the authors decided to select actions through sampling, their code also allows for deterministically selecting the best action. Note, all the deep neural networks discussed in this section are illustrated in Figure 1, and the neural networks introduced in Section 3 have similar architectures.

2.1.1 The Monte-Carlo tree search

Monte-Carlo tree search (MCTS) is used to enhance the planning ability of the DAI_{MC} 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 fed into the encoder to get the mean vector μ of the posterior over the latent states, i.e., $Q_{\phi_s}(s_t) = \mathcal{N}(s_t; \mu, I\sigma)$. Since μ is the mean of the Gaussian

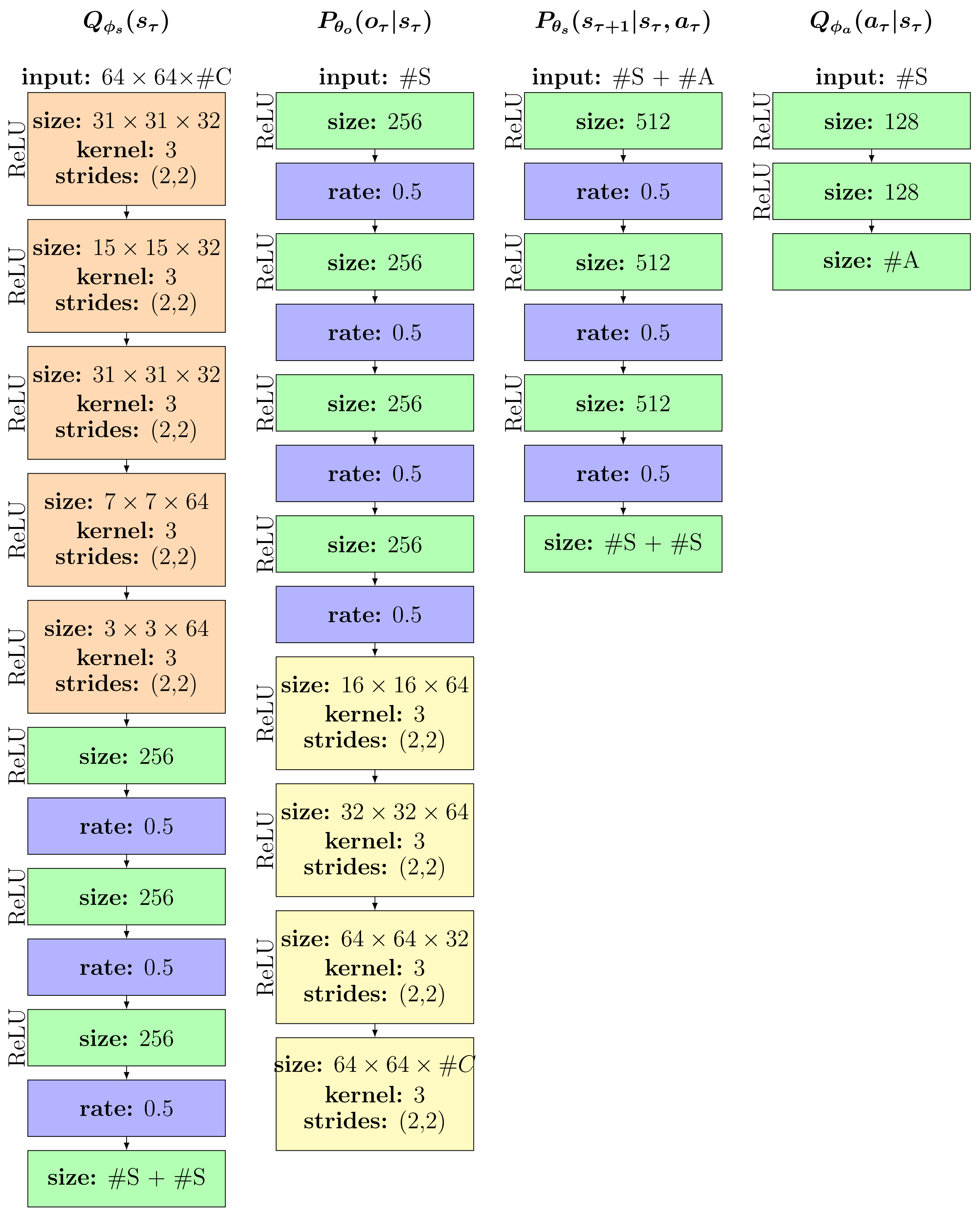

$Q_{\phi_s}(s_\tau)$
input: 64 × 64×#C
size: 31 × 31 × 32 kernel: 3 strides: (2,2) ReLU
size: 15 × 15 × 32 kernel: 3 strides: (2,2) ReLU
size: 31 × 31 × 32 kernel: 3 strides: (2,2) ReLU
size: 7 × 7 × 64 kernel: 3 strides: (2,2) ReLU
size: 3 × 3 × 64 kernel: 3 strides: (2,2) ReLU
size: 256 ReLU
rate: 0.5
size: 256 ReLU
rate: 0.5
size: 256 ReLU
rate: 0.5
size: #S + #S
$P_{\theta_o}(o_\tau|s_\tau)$
input: #S
size: 256 ReLU
rate: 0.5
size: 256 ReLU
rate: 0.5
size: 256 ReLU
rate: 0.5
size: 256 ReLU
rate: 0.5
size: 16 × 16 × 64 kernel: 3 strides: (2,2) ReLU
size: 32 × 32 × 64 kernel: 3 strides: (2,2) ReLU
size: 64 × 64 × 32 kernel: 3 strides: (2,2) ReLU
size: 64 × 64 × #C kernel: 3 strides: (2,2)
$P_{\theta_s}(s_{\tau+1}|s_\tau, a_\tau)$
input: #S + #A
size: 512 ReLU
rate: 0.5
size: 512 ReLU
rate: 0.5
size: 512 ReLU
rate: 0.5
size: #S + #S
$Q_{\phi_a}(a_\tau|s_\tau)$
input: #S
size: 128 ReLU
size: 128 ReLU
size: #A

Figure 1: 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.

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 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:

$$UCT(\hat{s}_\tau, \hat{a}_\tau) = -\bar{G}(\hat{s}_\tau, \hat{a}_\tau) + C_{explore} \cdot \frac{Q_{\phi_a}(a_\tau = \hat{a}_\tau | s_\tau = \hat{s}_\tau)}{1 + N(\hat{s}_\tau, \hat{a}_\tau)}, \tag{10}$$

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 (7), 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, and increase the number of visits by one.

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.

2.1.2 Minimization of the variational free energy

Champion et al (2023) provide a derivation for the variational free energy used by Fountas et al (2020), which is defined as follows:

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

By comparing the VFE in (11) and the EFE in (7), one can see an inconsistency. Namely, the parameters are seen as latent variables in the EFE definition, c.f., θ in (7), but they are regarded as parameters of neural networks in the VFE, c.f., θ_s and θ_o in 11. Note, θ cannot be both a parameter (i.e, parameter, vector, or matrix) and a random variable, and if θ is a random variable, one must define its probability density $P(\theta)$, which is missing in Equation 11. 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 optimized using back-propagation to minimize the VFE given by (11). Note, (11) 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. The EFE as stated in Equation (7) needs to be re-arranged because it cannot be easily evaluated. As shown in Champion et al (2023), Fountas et al (2020) made two independence assumptions without providing the proof, which questions the validity of the resulting EFE. Nonetheless, DAI_{MC} solves the dSprites environment.

2.2 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`. Although 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, that is, the code implements a *fully* observable decision process, i.e., MDP. Additionally, the DAI_{VPG} is composed of three neural networks. 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:

$$P(a_\tau|o_\tau) = \sigma[-\zeta G(o_\tau, a_\tau)], \tag{12}$$

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 model is based on the POMDP formalism. Therefore, $G(o_\tau, a_\tau)$ is denoted $G(s_\tau, a_\tau)$, and is defined as follows:

$$G(s_\tau, a_\tau) = -r_\tau + \underbrace{D_{\text{KL}}\left[\, Q(s_\tau) || \, Q(s_\tau|o_\tau)\right]}_{\text{intrinsic value}} + \hat{\mathcal{G}}_{\hat{\theta}_a}(a_{\tau+1}, s_{\tau+1}), \tag{13}$$

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 synchronized 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. Importantly, note that (13) is the definition of the expected free energy in the POMDP setting. As explained by Costa et al (2022), the expected free energy in the MDP setting is given by:

$$G(a_{t:T-1}, o_t) \approx \sum_{\tau=t+1}^{T} D_{\text{KL}}\left[\, P(o_\tau|a_{t:T-1}, o_t) || \, P(o_\tau)\right], \tag{14}$$

where $a_{t:T-1}$ is the sequence of actions between time steps t and $T-1$, $P(o_\tau)$ 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 decompose into extrinsic and intrinsic terms as in (13). 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 an MDP, and the expected free energy is adapted from the POMDP setting.

We conclude this section by discussing the training procedure of the transition, policy and critic networks. The transition network is trained to minimize the variational free energy, where the KL-divergence reduces to the mean squared error (MSE) because of two Gaussian assumptions (with identity covariance matrices), i.e., $\text{MSE}[o_{\tau+1}, \mathring{o}_{\tau+1}]$ where $\mathring{o}_{\tau+1} = \mathcal{T}_{\theta_o}(o_\tau, a_\tau)$ and $o_{\tau+1}$ is the observations made by the agent at time $\tau + 1$. The policy network is trained to minimize 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., $D_{\text{KL}}\left[\,Q_{\phi_a}(a_\tau|o_\tau)||\,P(a_\tau|o_\tau)\right]$, which minimizes the variational free energy. Finally, the critic is trained by minimizing the MSE between the target EFE as defined in (13) and the output of the critic $\mathcal{G}_{\theta_a}(o_\tau, \bullet)$, i.e., $\text{MSE}[G(o_\tau, \bullet), \mathcal{G}_{\theta_a}(o_\tau, \bullet)]$. DAI_{VPG} is able to solve the CartPole environment.

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

Rood et al (2020) 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.

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

In this section, we 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 minimization 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.

2.5 DAI_{FA} agent (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., $Q_{\phi_s}(s_t|s_{t-1}, o_t) = \mathcal{N}(s_t; \mu, I\sigma)$ 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., $P_{\theta_o}(o_\tau|s_\tau) = \mathcal{N}(o_\tau; \mu_o, I\sigma_o)$ 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., $P_{\theta_s}(s_\tau|s_{\tau-1}) = \mathcal{N}(s_\tau; \mathring{\mu}, I\mathring{\sigma})$ 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 is because the action is contained in the observations predicted by the decoder. 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, consistent with 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., $P_{\theta_a}(a_\tau|s_\tau) = \mathcal{N}(a_\tau; \mu_a, I\sigma_a)$ where $\mu_a, \sigma_a = \mathcal{P}_{\theta_a}(s_\tau)$. Then, Ueltzhöffer (2018) defines the free action (FA) objective as the cumulated variational free energy over time:

$$FA(o_{1:T},\phi,\theta) = \sum_{\tau=1}^{T}\Bigg[\underbrace{\overbrace{-\mathbb{E}_{Q_{\phi_s}(s_\tau|s_{\tau-1},o_\tau)}\Big[\ln P_{\theta_o}(o_\tau|s_\tau)\Big]}^{\text{accuracy}} + \overbrace{D_{\mathrm{KL}}\left[\,Q_{\phi_s}(s_\tau|s_{\tau-1},o_\tau)||\,P_{\theta_s}(s_\tau|s_{\tau-1})\right]}^{\text{complexity}}}_{\text{VFE}_\tau}\Bigg], \tag{15}$$

where $o_{1:T}$ is the sequence of observations between time step 1 and T (both included), $s_0 = (0,...,0)$ represents 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. The first action-perception cycle is initiated when the intital hidden state s_0 is 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_1^x. Next, the action $\hat{a}_0$ is concatenated with o_1^x 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 the current 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 also 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 minimize 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 for each new environment it is tested on. 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.

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

The code of the approach proposed by van der Himst and Lanillos (2020) is available here: `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, therefore: $P_{\theta_o}(o_\tau|s_\tau) = \mathcal{B}\text{ernoulli}(o_\tau; \hat{o}_\tau)$ 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., $P_{\theta_s}(s_{\tau+1}|s_\tau, a_\tau) = \mathcal{N}(s_{\tau+1}|\mathring{\mu}, I\mathring{\sigma})$ 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., $Q_{\phi_s}(s_t) = \mathcal{N}(s_t; \mu, I\sigma)$ 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 minimize the variational free energy defined as follows:

$$\begin{aligned} Q^*_\phi(s_t, a_t) &= \underset{Q_\phi(s_t,a_t)}{\arg\min}\, D_{\text{KL}}\left[\, Q_{\phi_a}(a_t|s_t)Q_{\phi_s}(s_t) \middle|\middle|\, P_{\theta_o}(o_t|s_t)P_{\theta_s}(s_t|s_{t-1}, a_{t-1})P_{\theta_a}(a_t|s_t)\right] \qquad (16) \\ &= \underset{Q_\phi(s_t,a_t)}{\arg\min}\, D_{\text{KL}}\left[\, Q_{\phi_s}(s_t) \middle|\middle|\, P_{\theta_s}(s_t|s_{t-1}, a_{t-1})\right] + D_{\text{KL}}\left[\, Q_{\phi_a}(a_t|s_t) \middle|\middle|\, P_{\theta_a}(a_t|s_t)\right] - \mathbb{E}_{Q_{\phi_s}(s_t)}[\ln P_{\theta_o}(o_t|s_t)]. \end{aligned}$$

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

$$Q^*_\phi(s_t, a_t) = \underset{Q_\phi(s_t,a_t)}{\arg\min}\, \text{MSE}(\mu, \mathring{\mu}) + D_{\text{KL}}\left[\, Q_{\phi_a}(a_t|s_t) \middle|\middle|\, P_{\theta_a}(a_t|s_t)\right] - \mathbb{E}_{Q_{\phi_s}(s_t)}[\ln P_{\theta_o}(o_t|s_t)], \qquad (17)$$

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.

That is, 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.

2.7 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. In the present paper, we focus on building an online approach that does not require manual gathering of the data.

2.8 Active exploration for robotic manipulation (Schneider et al, 2022)

The last paper that we review (Schneider et al, 2022) is motivated from a reinforcement learning perspective, where the agent aims to maximize reward. However, instead of greedily maximizing reward, the agent also maximizes the information gain between the model parameters and the expected states and rewards, i.e.,

$$\max_{\pi} \quad \underbrace{\mathbb{E}_{P(\boldsymbol{r}|\pi)}\big[f(\boldsymbol{r})\big]}_{\text{Expected reward}} + \beta \underbrace{\mathbb{E}_{P(\boldsymbol{s},\boldsymbol{r}|\pi,s_t)}\big[D_{\mathrm{KL}}\left[\, P(\theta|\boldsymbol{s},\boldsymbol{r},\pi,s_t) || \, P(\theta)\right]\big]}_{\text{Information gain}}, \quad \text{where:} \quad f(\boldsymbol{r}) = \sum_{\tau=t+1}^{t+H} r_\tau \quad (18)$$

where t is the current time step, H is the time horizon of planning, π is the agent policy, θ are the parameters of the model, $\boldsymbol{r} = \boldsymbol{r}_{t+1:t+H}$ is the sequence of rewards between time step $t+1$ and $t+H$, $\boldsymbol{s} = \boldsymbol{s}_{t+1:t+H}$ is the sequence of states between time step $t+1$ and $t+H$, and β is a hyper-parameter modulating the impact of information gain. Note, the distribution over the sequence of rewards $\boldsymbol{r}$ obtained by the agent when behaving according to a policy π, i.e., $P(\boldsymbol{r}|\pi)$, is not deterministic. Indeed, the same policy can produce different trajectories of states as the transition mapping is stochastic, and those different states can produce different rewards. Additionally, the expectation w.r.t $P(\boldsymbol{r}|\pi)$ is iterating over all possible sequences of rewards, and passing each sequence to $f(\boldsymbol{r})$. Then, the authors demonstrate the relationship between Equation 18 and the expected free energy. This relationship is explained in more details in (Schneider, N.D.), and relies on the following assumptions:

1. the states s_τ and rewards r_τ are latent variables for all $\tau \in \{t+1, ..., t+H\}$
2. the generative model contains observed variables for states o^s_τ and rewards o^r_τ
3. the likelihood mappings, i.e., $P(o^r_\tau|r_\tau)$ and $P(o^s_\tau|s_\tau)$, are delta distributions which effectively make the latent variables fully observable, except for the model parameters θ that are distributed according to a delta distribution at time step zero and remain fixed over time
4. the variance of the transition mapping $P(s_{\tau+1}|s_\tau, a_\tau, \theta)$ does not depend on the parameters θ, i.e., $P(s_{\tau+1}|s_\tau, a_\tau, \theta) = \mathcal{N}(s_{\tau+1}; \boldsymbol{\mu}, \boldsymbol{\sigma})$ where $\boldsymbol{\mu} = f_\theta(s_\tau, a_\tau)$ is predicted by a neural network with parameters θ, and $\boldsymbol{\sigma} = I\sigma$ is a diagonal matrix whose diagonal elements are equal to σ.

Importantly, while the above assumptions allow the authors to derive the expected free energy from Equation 18, these assumptions also impose a lot of constraints on the model. For example, the proof does not hold if the likelihood mappings are not delta distributions. In this paper, we focus on such more general models. To conclude, this approach is a great contribution to reinforcement learning applied to robotic control, but cannot be considered as a complete deep active inference agent. The code of this approach is available at the following URL: `https://github.com/TimSchneider42/aerm`.

3. Incrementally building a deep active inference agent

All 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 realization of deep active inference, because these approaches are either incomplete, unable to scale to image-based tasks, or make restrictive assumptions (see Section 2 for details). Accordingly, in the remainder of this paper, we step back to first principles, and "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 standard (well-attested) approaches: DQN, PPO, and A2C. Thus, in this section, we progressively build a deep active inference agent. Section 3.1 presents the dSprites environment in which all our simulations will be run. This environment was explored because of its use in (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, and finally, Section 3.5 presents a deep critical HMM (CHMM) agent. Importantly, building a VAE and a HMM first allows us to validate the implementation of individual components (i.e., encoder, decoder, and transition network), before tackling the CHMM, which represents a complete implementation of active inference. This method also allows us to better localize the source of potential problems.

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 (Higgins et al, 2017).

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 2.

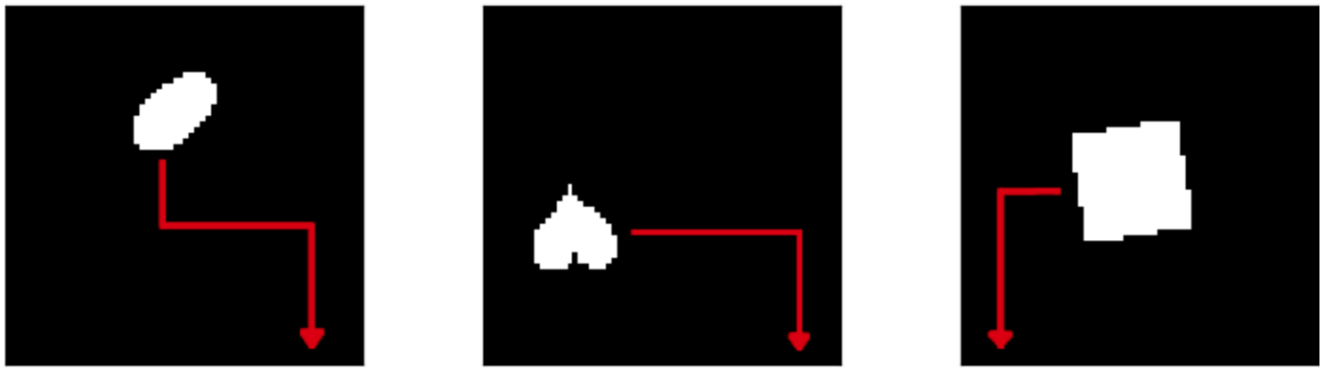


Figure 2: This figure illustrates the dSprites environment, in which the agent must move all squares towards the bottom-left corner of the image and all ellipses and hearts towards the bottom-right corner of the image. The red arrows show the behaviour expected from the agent.

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 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 **??** summarises the agent-environment interaction.

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 experience to 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, which represents the first stepping stone towards a deep active inference agent as it extracts latent variables from the images. The agent is composed of two deep neural networks, i.e., an encoder[3] and a decoder[4].

[3]The encoder is a convolutional neural network.

[4]The decoder is a deconvolutional neural network.

The encoder $\mathcal{E}_{\phi_s}$ takes as input an image o_t and outputs the parameters of the variational posterior $Q_{\phi_s}(s_t) = \mathcal{N}(s_t; \mu, I\sigma)$, 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:

$$P_{\theta_o}(o_t|s_t) = \mathcal{B}\text{ernoulli}(o_t; \hat{o}_t), \tag{19}$$

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:

$$\mathcal{B}\text{ernoulli}(o_t; \hat{o}_t) = \prod_{x,y} \text{Bernoulli}(o_t[x, y]; \hat{o}_t[x, y]), \tag{20}$$

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 minimize the variational free energy (VFE):

$$\boldsymbol{F} = D_{\text{KL}}\left[\, Q_{\phi_s}(s_t) || \, P_{\theta_o}(o_t, s_t)\right] = D_{\text{KL}}\left[\, Q_{\phi_s}(s_t) || \, P_{\theta_o}(o_t|s_t)P(s_t)\right], \tag{21}$$

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

$$\boldsymbol{F} = D_{\text{KL}}\left[\, Q_{\phi_s}(s_t) || \, P(s_t)\right] - \mathbb{E}_{Q_{\phi_s}(s_t)}[\ln P_{\theta_o}(o_t|s_t)], \tag{22}$$

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 reparameterization trick as follows: $\hat{s}_t = \mu + \sigma \odot \hat{\epsilon}$, where $\odot$ is an element-wise product between two vectors, and $\hat{\epsilon} \sim \mathcal{N}(\epsilon; 0, I)$.

This agent takes random actions, and stores its experiences in a 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 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 $Q_{\phi_s}(s_t) = \mathcal{N}(s_t; \mu, I\sigma)$. A latent state is sampled from $Q_{\phi_s}(s_t)$ using the reparameterization 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[5] 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 3 illustrates the VAE agent presented in this section. Note, this agent takes random actions.

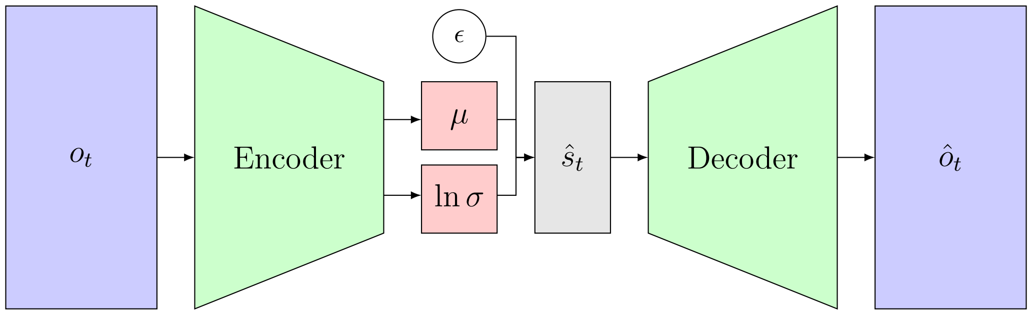


Figure 3: This figure illustrates the VAE agent. From left to right, we have the input image o_t, the encoder network, the layer of mean μ and log variance $\ln \sigma$, the epsilon random variable used for the reparameterization trick, the latent state $\hat{s}_t$, the decoder network, and finally, the reconstructed image $\hat{o}_t$. Note, there are no actions in this agent's model, thus, the VAE takes random actions.

3.4 Deep hidden Markov model

In this section, we present our second agent based on a hidden Markov model, which represents the second stepping stone toward a deep active inference agent. The HMM builds upon the latent variables extracted by the VAE, by modelling their temporal transition. More precisely, the HMM

[5]The VFE is equal to the negative evidence lower bound, i.e., $VFE = -ELBO$.

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[6] $\mathcal{T}_{\theta_s}$ that predicts the parameters of the Gaussian distribution $P_{\theta_s}(s_{t+1}|s_t, a_t) = \mathcal{N}(s_{t+1}; \mathring{\mu}, I\mathring{\sigma})$, where $\mathring{\mu}$ is the mean of the Gaussian distribution, and $\mathring{\sigma}$ are the diagonal elements of the covariance matrix. Additionally, $P(s_t) = \mathcal{N}(s_t; 0, I)$ is an isotropic (multivariate) Gaussian with variance one. Recall, that the goal of the inference process is to fit the approximate posterior $Q_{\phi_s}(s_{t+1}, s_t)$ to the true posterior $P(s_{t+1}, s_t|o_{t+1}, o_t, a_t)$. Formally, this optimization can be written as the following minimization:

$$Q^*_{\phi_s}(s_{t+1}, s_t) = \underset{Q_{\phi_s}(s_{t+1},s_t)}{\arg\min}\; D_{\mathrm{KL}}\left[\, Q_{\phi_s}(s_{t+1}, s_t) || \, P(s_{t+1}, s_t|o_{t+1}, o_t, a_t)\right]. \tag{23}$$

Using a derivation almost identical to the one presented in Section 2.1.2, the VFE can be proven to be:

$$\begin{aligned} Q^*_{\phi_s}(s_{t+1}, s_t) \;&=\; \underset{Q_{\phi_s}(s_{t+1},s_t)}{\arg\min}\; \underbrace{D_{\mathrm{KL}}\left[\, Q_{\phi_s}(s_{t+1}, s_t) || \, P_{\theta_o}(o_t|s_t)P_{\theta_o}(o_{t+1}|s_{t+1})P_{\theta_s}(s_{t+1}|s_t, a_t)P_{\theta_s}(s_t)\right]}_{\text{variational free energy}} \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (24) \\ &=\; \underset{Q_{\phi_s}(s_{t+1},s_t)}{\arg\min}\; D_{\mathrm{KL}}\left[\, Q_{\phi_s}(s_{t+1}) || \, P_{\theta_s}(s_{t+1}|s_t, a_t)\right] - \mathbb{E}_{Q_{\phi_s}(s_t)}[P_{\theta_o}(o_{t+1}|s_{t+1})] \\ &\qquad\qquad + D_{\mathrm{KL}}\left[\, Q_{\phi_s}(s_t) || \, P_{\theta_s}(s_t)\right] - \mathbb{E}_{Q_{\phi_s}(s_t)}[P_{\theta_o}(o_t|s_t)]. \end{aligned}$$

The VFE of Equation 24 can be computed in a similar way to the VAE agent. Put simply, this agent takes random actions, and stores its experiences in a 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 replay buffer. The observations at time step t and $t+1$ are fed into the encoder, which outputs the mean and log variance of a Gaussian distribution $Q_{\phi_s}(s_t) = \mathcal{N}(s_t; \mu, I\sigma)$, and $Q_{\phi_s}(s_{t+1}) = \mathcal{N}(s_{t+1}; \hat{\mu}, I\hat{\sigma})$, respectively. A latent state is sampled from $Q_{\phi_s}(s_t)$ using the reparameterization trick, and is then provided as input to the transition network along with action a_t. The transition network outputs the parameters of the

[6]The transition network is a fully connected neural network.

Gaussian distributions $P_{\theta_s}(s_{t+1}|s_t, a_t) = \mathcal{N}(s_{t+1}; \mathring{\mu}, I\mathring{\sigma})$. Additionally, the observations at time step t can be fed into the encoder to get the parameters of $Q_{\phi_s}(s_t) = \mathcal{N}(s_t; \hat{\mu}, I\hat{\sigma})$. Sampling from $Q_{\phi_s}(s_t)$ and $Q_{\phi_s}(s_{t+1})$ using the reparameterization trick gives two states $\hat{s}_t$ and $\hat{s}_{t+1}$ that when given as input to the decoder produces the parameters of a product of Bernoulli distributions $\hat{o}_t$ and $\hat{o}_{t+1}$. The KL-divergences between $Q_{\phi_s}(s_t)$ and $P_{\theta_s}(s_t)$, as well as between $Q_{\phi_s}(s_{t+1})$ and $P_{\theta_s}(s_{t+1}|s_t, a_t)$ are computed analytically, and the logarithm of $P_{\theta_o}(o_\tau|s_\tau)$ reduces to the binary cross entropy (BCE) because $P_{\theta_o}(o_\tau|s_\tau)$ is a product of Bernoulli distributions. Next, the VFE is obtained by subtracting the BCE at time t and $t+1$ from the two KL-divergences, and back-propagation is used to update the weights of the encoder, decoder and transition networks. Figure 4 illustrates the HMM agent.

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 represents a complete implementation of active inference, c.f., Figure 5. 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 $P_{\theta_s}(s_{t+1}|s_t, a_t)$, and a critic network[7] $\mathcal{G}_{\theta_a}$ 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 minimize the VFE. The critic however is trained to minimize 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 minimizes SL1$[\mathcal{G}_{\theta_a}(s_t, \bullet), y(\bullet)]$. Note, the SL1 was picked because it is less sensitive to outliers than

[7]The critic network is a fully connected neural network.

the MSE, and is defined as:

$$SL1[x, y] = \begin{cases} 0.5 \times \frac{(x-y)^2}{\beta} & \text{if } |x - y| < \beta \\ |x - y| - 0.5 \times \beta & \text{otherwise} \end{cases}, \tag{25}$$

where β is an hyper-parameter such that as β goes to zero, the SL1 loss converges to the L1 loss, and when β tends to $+\infty$, the SL1 loss converges to a constant zero loss. Intuitively, the SL1 loss uses a squared term if the absolute element-wise error falls below β, and an L1 term otherwise. Additionally, the target G-values are defined as:

$$y(a_t) = G_{t+1}(a_t) + \gamma \mathbb{E}_{Q_{\phi_s}(s_{t+1})}\Big[\max_{a_{t+1} \in \mathcal{A}} \hat{\mathcal{G}}_{\hat{\theta}_a}(s_{t+1}, a_{t+1})\Big], \tag{26}$$

where $\mathcal{A}$ is the set of all actions, and $Q_{\phi_s}(s_{t+1})$ can be computed by feeding the image o_{t+1} sampled from the 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 of Bellman's equation (Bellman, 1952) to the expected free energy. Thus, in contrast to the standard active inference objective that is not Bellman-optimal for time horizons greater than one (Da Costa et al, 2023), our approach is Bellman-optimal and closely resembles sophisticated inference (Friston et al, 2020). 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 synchronized 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 during training?

There are at least four possibilities: (i) select a random action, (ii) select the action that minimizes EFE according to the critic, i.e., $a_t^* = \arg\min_{a_t} \mathcal{G}_{\theta_a}(s_t, a_t)$, (iii) sample an action from a softmax function of (minus) the critic's output, 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.

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 (2019) defined the expected free energy as:

$$G(\pi) = \sum_{\tau=t+1}^{T} G_\tau(\pi) = \sum_{\tau=t+1}^{T} \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}\big[\ln Q(s_\tau|\pi) - \ln P(o_\tau, s_\tau)\big], \tag{27}$$

where π is a sequence of actions, $P(o_\tau|s_\tau)$ is the likelihood mapping, $Q(s_\tau|\pi)$ is the variational distribution, and 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 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. According to the free energy principle, an active inference agent must minimize its (variational) free energy. However, if such a relationship cannot be established between the expected and variational free energy, then one cannot claim that an agent minimizing expected free energy also minimizes its variational free energy. Additionally, we need to re-arrange the definition of the EFE stated in (27) to allow rewards to be incorporated, i.e., as seen in the extrinsic value term:

$$\begin{aligned} G_\tau(\pi) &= \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}\big[\ln Q(s_\tau|\pi) - \ln P(o_\tau, s_\tau)\big] \\ &= \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}\big[\ln Q(s_\tau|\pi) - \ln P(s_\tau|o_\tau) - \ln P(o_\tau)\big] \\ &\approx \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}\big[\ln Q(s_\tau|\pi) - \ln Q(s_\tau) - \ln P(o_\tau)\big] \\ &= \underbrace{D_{\mathrm{KL}}\left[\, Q(s_\tau|\pi) || \, Q(s_\tau)\right]}_{\text{epistemic value}} - \underbrace{\mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}\big[\ln P(o_\tau)\big]}_{\text{extrinsic value}}. \end{aligned} \tag{28}$$

The above EFE formulation was proposed by Parr and Friston (2019), and has become the typical EFE definition in deep active inference. However, there are alternative definitions, which are likely to behave better (Parr et al, 2022). Importantly, there remain questions about the mathematical principle underlying the EFE definition, and this is a subject of ongoing work in our group (Champion et al, 2024). As shown in Appendix A, Equation 28 can be approximated as follows:

$$G_{t+1}(a_t) \approx D_{\text{KL}} \left[P_{\theta_s}(s_{t+1}|s_t = \hat{s}_t, a_t) || \, Q_{\phi_s}(s_{t+1}) \right] - \psi r_{t+1}, \tag{29}$$

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 6, the reward at time step $t+1$ is used to compute the target values that must be predicted by the critic network.

3.5.2 Maximizing reward 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. Therefore, we experimented with simply predicting the (negative) expected future reward as follows:

$$G^r_{t+1}(a_t) = -\psi r_{t+1}, \tag{30}$$

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

4. Results

In this section, we discuss the results obtained by each model presented in Section 3, as well as three baseline agents (i.e., DQN, PPO, and A2C) when solving the dSprites problem. However, we did not compare with the methods presented in Section 2 because these approaches are either incomplete, unable to scale to image-based tasks, or make restrictive assumptions. The code that can be used to reproduce all the experiments can be found on GitHub at the following URL: `https://github.com/ChampiB/Deconstructing_Deep_Active_Inference`.

4.1 Variational free energy and reconstructed images

To check the validity of our implementation, we first look at the variational free energy (VFE) obtained by the VAE, HMM and CHMM agents. As shown in Figure 7, all the agents were able to minimize their variational free energy, and the VFE smoothly decreases during training. By comparing the curves in Figure 7, one can see that the HMM and CHMM have a higher VFE than the VAE. This is because the VFE of the HMM and CHMM agents is composed of two KL divergences and two negative log-likelihood instead of one. In addition, we plotted the images reconstructed by each agent, c.f., Figure 8 for VAE agent, Figure 9 for HMM agent, and Figure 10 for CHMM agent. After 500K training iterations, all the agents were able to properly generate images or sequences of images (see below).

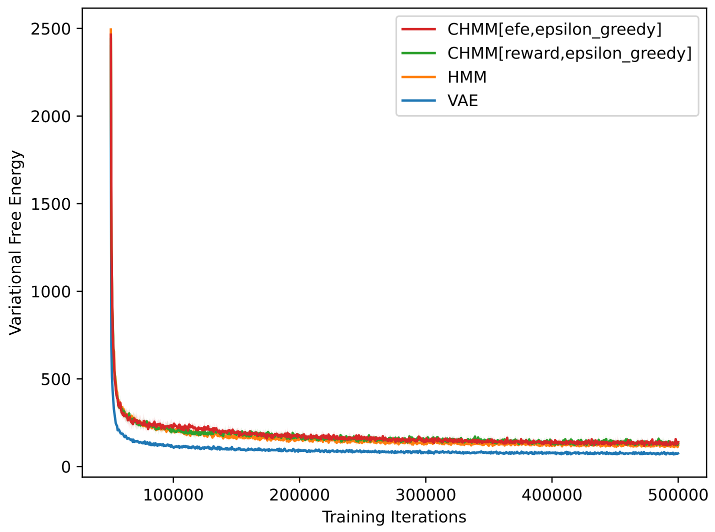


Figure 7: This figure illustrates the variational free energy of the VAE, HMM, and the CHMM agents during the 500K iterations of training. The CHMMs were maximizing reward and expected free energy using $\mathring{\epsilon}$-greedy action selection.

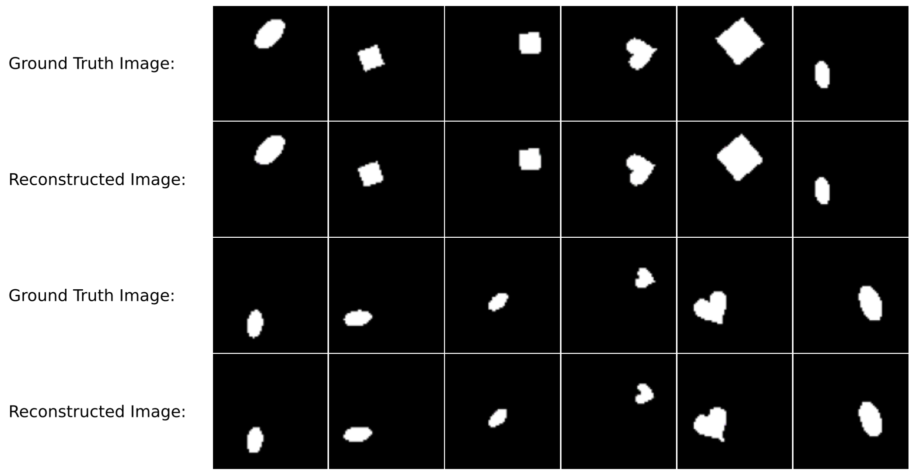


Figure 8: This figure illustrates the images generated by the VAE after 500K training iterations.

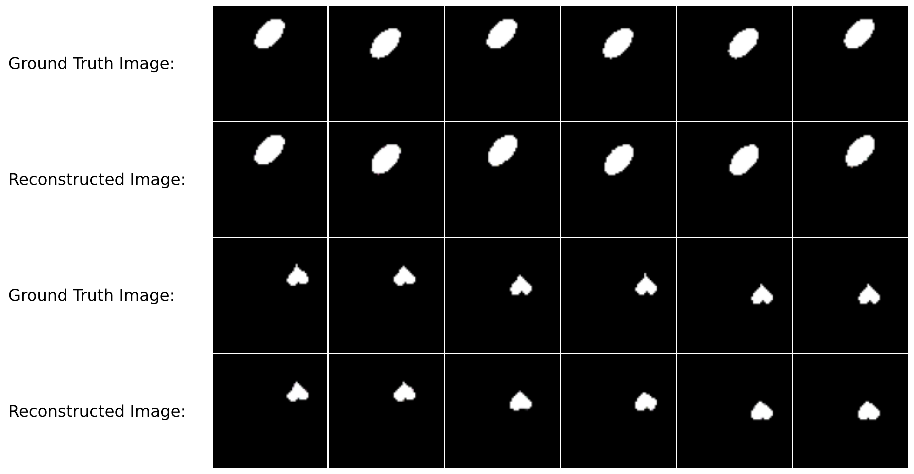


Figure 9: This figure illustrates the sequences of reconstructed images generated by the HMM after 500K training iterations. Time passes horizontally (from left to right).

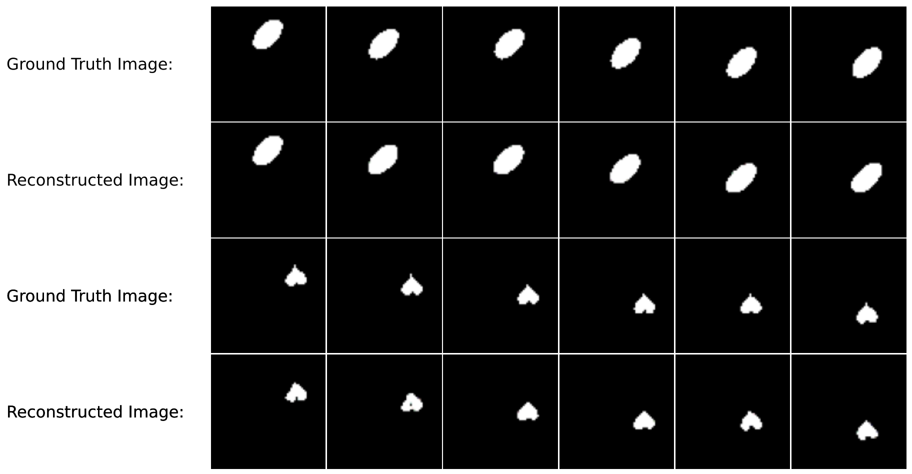


Figure 10: This figure illustrates the sequences of reconstructed images generated by a CHMM (maximizing reward) after 500K training iterations. Time passes horizontally (from left to right).

Importantly, for the VAE, the reconstructed images were obtained by feeding the ground truth images into the encoder, and presenting the mean outputted by the encoder as input to the decoder. Similarly, for the HMM and CHMM, the first ground truth image was provided as input to the encoder, which outputs the mean used as input to the decoder to create the first reconstructed image. The mean is also presented as input to the transition network alongside an action to generate the mean at the next time step, which is fed to the decoder to obtain the second reconstructed image. This process continues to generate the subsequent reconstructed images.

4.2 Rewards obtained by each agent

Note, a random agent is as likely to cross the bottom border on the right and left side of the image. Thus on average the agent will receive zero reward when crossing the bottom border. In contrast, if the agent does not cross the bottom line within the first 50 actions, it will receive a reward of -1. Thus, a random agent is expected to gather a negative amount of reward. Since the VAE and HMM agents take random actions in the environment, the agents were unable to solve the task and accumulated a total amount of reward of around -7K (after 500K training iterations). These results suggest that our implementation is correct, and gives us a baseline for the amount of rewards obtained under random play in the dSprites environment.

Figure 11 shows the mean episodic reward gathered (during 500K training iterations) by the CHMM agents with (a) $\mathring{\epsilon}$-greedy, (b) softmax, and (c) best action selection. The CHMM performance is compared to the stable baseline implementation of PPO, A2C and DQN with default parameters. A2C and DQN obtained about the same asymptotic reward, and were outperformed by PPO. Also, A2C initially learns much faster than DQN, but suffers from a decrease in performance between 180K and 280K training iterations. In contrast, DQN has a slower but more stable learning on this task.

The CHMM maximizing reward outperformed the other models when using $\mathring{\epsilon}$-greedy action selection, but was unable to solve the task with softmax action selection. Moreover, when using best action selection, the CHMM maximizing reward performed as well as PPO, but had higher variance. Also, by comparing Figures 11(a) and 11(c), 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. Importantly, the CHMM minimizing EFE was unable to solve the task independently of the action selection strategy.

4.2.1 Degenerate behaviour with the expected free energy?

Up to now, we saw that the CHMM minimizing expected free energy (EFE) was not able to solve the task. To investigate why, we looked at the action selected by the CHMM agents, c.f., Figure 13. Indeed, Figure 13(a) shows that the agent minimizing EFE learn to select almost exclusively the action down, and Figure 13(b) shows that the entropy of the prior over actions very quickly converges to zero.

In contrast, the CHMM maximizing reward, keeps on selecting the actions right and left, which enables it to drag the shape towards the appropriate corner, c.f., Figure 13(c). Also, as shown in Figure 13(d), the entropy of the prior over actions remained a lot higher than zero. Note, the only difference between the CHMM minimizing EFE and the one maximizing reward is the information gain (see Equation 28), which is defined as the KL divergence between the output of the transition and encoder networks. Since the EFE is minimized, the output of those two networks need to be as close as possible to each other.

This suggests that the CHMM minimizing 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 receives 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.

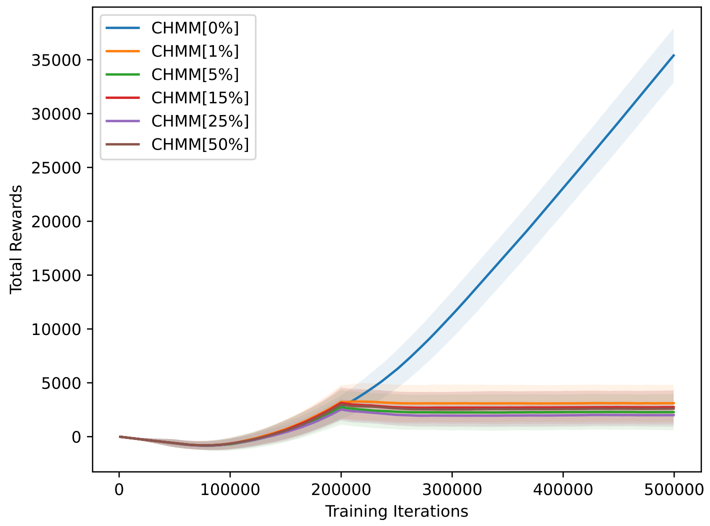
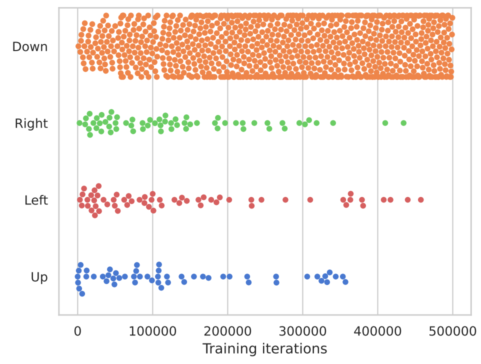
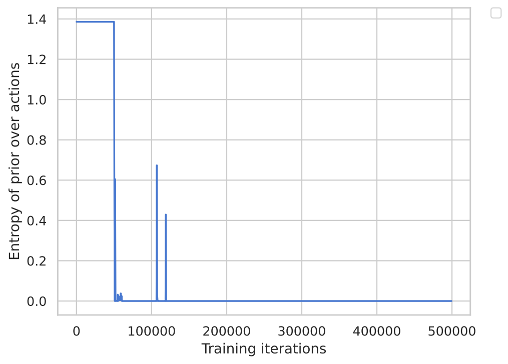


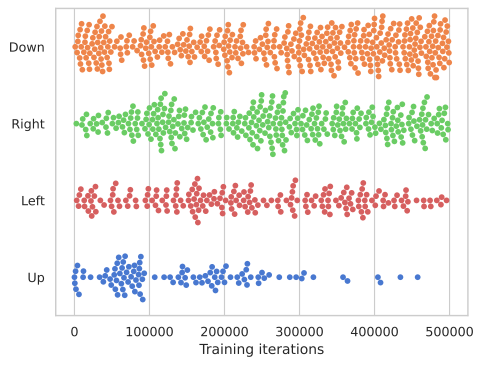
Figure 12: This figure illustrates the total reward aggregated by CHMM agents during the 500K iterations of training. All the agents start by only maximizing reward, and after 200K training iterations, X% of the information gain is added to the objective function. Note, even adding 1% of the information 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.



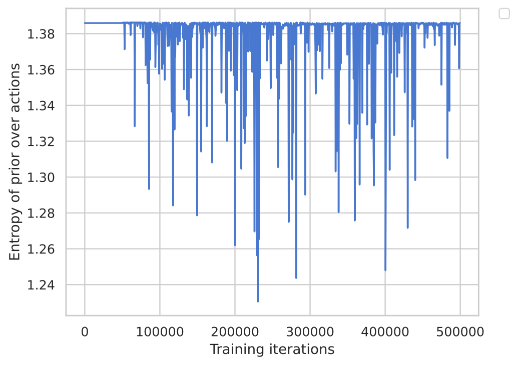
(a)



(b)



(c)



(d)

Figure 13: (a) and (c) show the action taken (for each training iteration) by the CHMM minimizing EFE and maximizing reward, respectively. (b) and (d) show the entropy of the prior over actions for the CHMM minimizing EFE and maximizing reward, respectively.

Also, we developed a new agent to shows the impact of adding X% of the information gain into the objective function, i.e., the agent starts by only maximizing reward (c.f. Equation 30), and after 200K training iterations minimizes reward plus X% of the information gain. As shown in Figure 12, adding even 1% of the information gain already dramatically decreases the amount of reward gathered.

To conclude, the same information gain that is intended to give an EFE minimizing 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. In Champion et al (2023), we experimented with another version of deep active inference agent. However, the

reward coefficient	objective	CHMM	DCHMM
1	reward	0.99	**1.0**
	EFE	-0.02	**0.09**
1000	reward	0.76	**1.0**
	EFE	0.56	**0.94**
2000	reward	0.63	**0.99**
	EFE	0.66	**0.97**
5000	reward	0.52	**0.97**
	EFE	0.47	**0.96**

Table 1: This table shows the performance of the CHMM[reward] and CHMM[EFE] with $\mathring{\epsilon}$-greedy, when the precision of the prior preferences ψ (reward coefficient for short) is set to 1, 1000, 2000, and 5000. Importantly, as ψ increases the CHMM[reward] and CHMM[EFE] performance becomes more similar. However, since the CHMM[reward] performance decreased as ψ increased, we created a DCHMM agent by adding layers to the CHMM's critic network.

results are not presented here because similarly to the simpler version, i.e., the CHMM[EFE], it failed to solve the task.

Finally, to validate our implementation of the CHMM[EFE], we increased the precision of the prior preferences ψ, and compared the performance to the reward maximising agent. As shown in Table 1, as ψ increases the performance of the CHMM[reward] and CHMM[EFE] agents become more similar. However, the CHMM[reward] performance decreased as ψ increased, thus, we created a CHMM with a deeper critic (DCHMM) by adding layers to the CHMM's critic network. This more powerful critic network was able to properly predict the scaled rewards, i.e., ψr_τ. Put simply, Table 1 demonstrates that as the information gain becomes small w.r.t. the reward term, the CHMM[EFE] and CHMM[reward] converge to the same behaviour.

5. Discussion of epistemic value

In this section, we discuss the decomposition of the expected free energy into epistemic and extrinsic value as given in Equation (10) of Parr and Friston (2019). A particular reason for being interested in this formulation is that it is the version of the expected free energy that has most influenced the deep active inference approaches, such as Fountas et al (2020). Starting with the definition of the expected free energy from Equation (28):

$$\begin{aligned} G_\tau(\pi) &= \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}\big[\ln Q(s_\tau|\pi) - \ln P(o_\tau, s_\tau)\big] \\ &= \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}\big[\ln Q(s_\tau|\pi) - \ln P(s_\tau|o_\tau) - \ln P(o_\tau)\big] \\ &= -\underbrace{\mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}\big[\ln P(s_\tau|o_\tau) - \ln Q(s_\tau|\pi)\big]}_{\text{Epistemic value}} - \underbrace{\mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}\big[\ln P(o_\tau)\big]}_{\text{Extrinsic value}}. \end{aligned} \tag{31}$$

More precisely, due to the poor results of the CHMM[EFE] agent, we seek to understand the damaging impact of the epistemic value on performance. Assuming the following definition for the epistemic value:

$$\boldsymbol{EV} = \mathbb{E}_{P(o_\tau|s_\tau)Q(s_\tau|\pi)}\big[\ln P(s_\tau|o_\tau) - \ln Q(s_\tau|\pi)\big], \tag{32}$$

we set up two experiments in which all the distributions are stored in tables. In other words, the following categorical distributions are represented using matrices: $P(o_\tau|s_\tau)$, $Q(s_\tau|\pi)$, and $P(s_\tau|o_\tau)$.

In the first experiment, the prior over states (c.f., left-most graph in Figure 18) is equal to the approximate posterior (c.f., Figure 15), and the likelihood (c.f., Figure 16) is becoming more and more uniform (c.f., Figure 17). Note, if the likelihood becomes uniform, then the true posterior $P(s_\tau|o_\tau)$ becomes equal to the prior over states $P(s_\tau)$, i.e.,

$$P(s_\tau|o_\tau) \propto P(o_\tau|s_\tau)P(s_\tau) = \frac{1}{|o_\tau|}P(s_\tau), \tag{33}$$

where $|o_\tau|$ is the number of observations at time step τ, and after renormalization $P(s_\tau|o_\tau) = P(s_\tau)$. The left-most graph of Figure 14 shows that the epistemic value decreases as the likelihood becomes more uniform, i.e., the epistemic value decreases as the true posterior $P(s_\tau|o_\tau)$ becomes more similar to the approximate posterior $Q(s_\tau|\pi)$. This behaviour is to be expected, as the epistemic value is bigger when the true posterior $P(s_\tau|o_\tau)$ and the approximate posterior $Q(s_\tau|\pi)$ are more different. Thus, maximizing epistemic value will promote exploration and information gain.

In the second experiment, the likelihood has high entropy (c.f., right-most graph in Figure 17), and the prior over states is shifting from left to right (c.f., Figure 18). When the prior over states $P(s_\tau)$ and the approximate posterior $Q(s_\tau|\pi)$ are different, the joint distribution $P(o_\tau|s_\tau)Q(s_\tau|\pi)$ will be more similar to the approximate posterior $Q(s_\tau|\pi)$ than it is to the true posterior $P(s_\tau|o_\tau)$. Indeed, as the likelihood is almost uniform, the true posterior $P(s_\tau|o_\tau)$ will be almost equal to the prior over states $P(s_\tau)$. At the same time, as the likelihood is almost uniform, it will not have much impact on the joint distribution $P(o_\tau|s_\tau)Q(s_\tau|\pi)$ and the approximate posterior $Q(s_\tau|\pi)$ will dominate. To sum up, when the prior over states $P(s_\tau)$ and the approximate posterior $Q(s_\tau|\pi)$ are different:

- the true posterior will almost be equal to the prior over states $P(s_\tau|o_\tau) \approx P(s_\tau)$
- the joint distribution $P(o_\tau|s_\tau)Q(s_\tau|\pi)$ will be more similar to $Q(s_\tau|\pi)$ than it is to $P(s_\tau|o_\tau)$

Therefore, the joint distribution $P(o_\tau|s_\tau)Q(s_\tau|\pi)$ will tend to be large when the difference within the expectation is negative (c.f., Figure 19). Indeed, as the joint is similar to the approximate posterior, it means that the joint distribution is large when the approximate posterior is large and the true posterior is smaller. This implies that the logarithm of true posterior will be very negative, while the logarithm of the approximate posterior will be less negative, i.e., closer to zero. Then, the logarithm of the approximate posterior will be subtracted from the logarithm of the true posterior, i.e., a small positive number will be added to a very negative number. Therefore, the result will be negative.

The right-most graph of Figure 14 shows that the epistemic value increases as the prior $P(s_\tau)$ and therefore the true posterior $P(o_\tau|s_\tau)$ becomes more similar to the approximate posterior $Q(s_\tau|\pi)$. This behaviour should not be observed, as the epistemic value is bigger when the true posterior $P(s_\tau|o_\tau)$ and the approximate posterior $Q(s_\tau|\pi)$ are more similar. Thus, maximizing epistemic value will not promote exploration. In fact, it will recommend a strong focus on a single action as was observed in Figure 13.

To sum up, the expected free energy decomposition into epistemic and extrinsic value — as presented in Equation (10) of Parr and Friston (2019) — seems to exhibit two very different behaviours depending on how the distributions are defined, c.f., Figure 14. This is particularly important in the deep active inference literature, which builds on this equation. For example, the graphs presented in Section 4.2.1 indicate that the CHMM agent minimizing expected free energy is focusing almost exclusively on the action "down", i.e., it is not exploring, which leads to poor performance.

6. 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. However, we did not compare with these methods as they are either incomplete, unable to scale to image-based tasks, or make restrictive assumptions.

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 we were unable to access code in some instances. Ideally, future research should draw inspiration from the open science framework, by making the code that produced the claimed results open source, and 100% compatible with the paper. 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 (c.f., Section 5). 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, and reproducibilty of research. For more details, the reader is referred to Section 2 and Champion et al (2023).

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 the expected free energy, and as the CHMM minimizing EFE was not able to solve the dSprites environment, we also tried to remove the information gain (simply predicting the reward). Additionally, we implemented three types of action selection strategies, namely: best action according to the critic, softmax sampling, and epsilon-greedy.

When the epsilon-greedy algorithm was used, only the agent maximizing reward was able to solve the task. When softmax sampling was used, all the agents failed to solve the task. Lastly, when selecting the best action, only the reward maximizing 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.

Finally, we compared the action selected by the CHMM minimizing EFE and the one maximizing reward. Visualizing the distribution of actions selected as training progresses shows that the agent minimizing EFE almost exclusively picks action down. In contrast, the CHMM maximizing 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 minimizing 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 when only one action is picked by the policy. Therefore, if the CHMM has to stick to a single action to keep the KL divergence small, then the action down should be selected as it is the most rewarding.

The CHMMs that maximizes reward and uses softmax sampling for action selection, cannot solve the task, and we can hypothesize 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, our investigations of the expected free energy often used in deep active inference (Fountas et al, 2020), suggests that degenerate behaviour can arise from it, in certain situations. This could explain why adding epistemic value to our planning objective seems to have such a damaging impact on the agent's capacity to explore its environment and gain information. The use of this definition of the expected free energy in the deep learning context may explain why the CHMM minimizing EFE was unable to solve the task, and may also explain some of the presentational uncertainties (e.g., additions of minus signs) found in the deep active inference literature, see Champion et al (2023) for details.

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 critic network and (optionally) a policy network, (ii) a thorough mathematical treatment is present, which is consistent with the free energy principle, (iii) the implementation is 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

We thank Karl J. Friston, Thomas Parr, Lancelot Da Costa, and Zafeirios Fountas for useful discussions and feedback surronding this paper, as well as for pointing us towards important resources such as their new book and several papers. We would also like to more broadly thank attendees at Karl Friston's Theory of Neurobiology group, who have provided valuable insights in response to our presentations. Finally, we thank the reviewers for their valuable feedback and insightful comments, which greatly improved the quality of the present paper.

References

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

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

Çatal O, Wauthier S, De Boom C, Verbelen T, Dhoedt B (2020) Learning generative state space models for active inference. Frontiers in Computational Neuroscience 14:574,372

Champion T, Grześ M, Bowman H (2021) Realizing Active Inference in Variational Message Passing: The Outcome-Blind Certainty Seeker. Neural Computation 33(10):2762–2826, DOI 10.1162/neco_a_01422, URL https://doi.org/10.1162/neco_a_01422, https://direct.mit.edu/neco/article-pdf/33/10/2762/1963368/neco_a_01422.pdf

Champion T, Bowman H, Grześ M (2022a) Branching time active inference: Empirical study and complexity class analysis. Neural Networks 152:450–466, DOI https://doi.org/10.1016/j.neunet.2022.05.010, URL https://www.sciencedirect.com/science/article/pii/S0893608022001824

Champion T, Da Costa L, Bowman H, Grześ M (2022b) Branching time active inference: The theory and its generality. Neural Networks 151:295–316, DOI https://doi.org/10.

1016/j.neunet.2022.03.036, URL https://www.sciencedirect.com/science/article/pii/S0893608022001149

Champion T, Grześ M, Bowman H (2022c) Branching Time Active Inference with Bayesian Filtering. Neural Computation 34(10):2132–2144, DOI 10.1162/neco_a_01529, URL https://doi.org/10.1162/neco_a_01529, https://direct.mit.edu/neco/article-pdf/34/10/2132/2042425/neco_a_01529.pdf

Champion T, Grześ M, Bowman H (2022d) Multi-modal and multi-factor branching time active inference. DOI 10.48550/ARXIV.2206.12503, URL https://arxiv.org/abs/2206.12503

Champion T, Grześ M, Bonheme L, Bowman H (2023) Deconstructing deep active inference. URL https://arxiv.org/abs/2303.01618

Champion T, Bowman H, Marković D, Grześ M (2024) Reframing the expected free energy: Four formulations and a unification. 2402.14460

Costa LD, Parr T, Sajid N, Veselic S, Neacsu V, Friston K (2020) Active inference on discrete state-spaces: a synthesis. URL https://arxiv.org/abs/2001.07203

Costa LD, Sajid N, Parr T, Friston K, Smith R (2022) Reward maximisation through discrete active inference. URL https://arxiv.org/abs/2009.08111

Cullen M, Davey B, Friston KJ, Moran RJ (2018) Active inference in openai gym: A paradigm for computational investigations into psychiatric illness. Biological Psychiatry: Cognitive Neuroscience and Neuroimaging 3(9):809 – 818, 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

Da Costa L, Sajid N, Parr T, Friston K, Smith R (2023) Reward Maximization Through Discrete Active Inference. Neural Computation 35(5):807–852, DOI 10.1162/neco_a_01574, URL https://doi.org/10.1162/neco_a_01574, https://direct.mit.edu/neco/article-pdf/35/5/807/2079473/neco_a_01574.pdf

Doersch C (2016) Tutorial on variational autoencoders. URL https://arxiv.org/abs/1606.05908

FitzGerald THB, Dolan RJ, Friston K (2015) Dopamine, reward learning, and active inference. Frontiers in Computational Neuroscience 9:136, DOI 10.3389/fncom.2015.00136, URL https://www.frontiersin.org/article/10.3389/fncom.2015.00136

Fountas Z, Sajid N, Mediano PAM, Friston K (2020) Deep active inference agents using Monte-Carlo methods. URL https://arxiv.org/abs/2006.04176

Friston K, FitzGerald T, Rigoli F, Schwartenbeck P, Doherty JO, Pezzulo G (2016) Active inference and learning. Neuroscience & Biobehavioral Reviews 68:862 – 879, DOI https://doi.org/10.1016/j.neubiorev.2016.06.022

Friston K, Costa LD, Hafner D, Hesp C, Parr T (2020) Sophisticated inference. 2006.04120

van Hasselt H, Guez A, Silver D (2015) Deep reinforcement learning with double Q-learning. URL https://arxiv.org/abs/1509.06461

Higgins I, Matthey L, Pal A, Burgess C, Glorot X, Botvinick M, Mohamed S, Lerchner A (2017) 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, URL https://openreview.net/forum?id=Sy2fzU9gl

van der Himst O, Lanillos P (2020) Deep active inference for partially observable mdps. CoRR abs/2009.03622, URL https://arxiv.org/abs/2009.03622, 2009.03622

Itti L, Baldi P (2009) Bayesian surprise attracts human attention. Vision Research 49(10):1295 – 1306, 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

Kingma DP, Welling M (2014) Auto-Encoding Variational Bayes. In: International Conference on Learning Representations, Banff, Canada, vol 2, URL http://arxiv.org/abs/1312.6114

Lample G, Chaplot DS (2016) Playing FPS games with deep reinforcement learning. 1609.05521

Lanillos P, Cheng G, et al (2020) Robot self/other distinction: active inference meets neural networks learning in a mirror. arXiv preprint arXiv:200405473

Matthey L, Higgins I, Hassabis D, Lerchner A (2017) dsprites: Disentanglement testing sprites dataset. https://github.com/deepmind/dsprites-dataset/

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

Millidge B (2020) Deep active inference as variational policy gradients. Journal of Mathematical Psychology 96:102,348, DOI https://doi.org/10.1016/j.jmp.2020.102348, URL http://www.sciencedirect.com/science/article/pii/S0022249620300298

Mnih V, Kavukcuoglu K, Silver D, Graves A, Antonoglou I, Wierstra D, Riedmiller M (2013) Playing atari with deep reinforcement learning. URL https://arxiv.org/abs/1312.5602

Mnih V, Badia AP, Mirza M, Graves A, Lillicrap TP, Harley T, Silver D, Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning. URL https://arxiv.org/abs/1602.01783

Oliver G, Lanillos P, Cheng G (2019) Active inference body perception and action for humanoid robots. arXiv preprint arXiv:190603022

Parr T, Friston KJ (2019) Generalised free energy and active inference. Biological cybernetics 113(5-6):495–513

Parr T, Pezzulo G, Friston KJ (2022) Active Inference: The Free Energy Principle in Mind, Brain, and Behavior. The MIT Press, DOI 10.7551/mitpress/12441.001.

0001, URL https://doi.org/10.7551/mitpress/12441.001.0001, https://direct.mit.edu/book-pdf/2246566/book_9780262369978.pdf

Pezzato C, Hernandez C, Wisse M (2020) Active inference and behavior trees for reactive action planning and execution in robotics. URL https://arxiv.org/abs/2011.09756

Rezende DJ, Mohamed S, Wierstra D (2014) Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In: Xing EP, Jebara T (eds) Proceedings of the 31st International Conference on Machine Learning, PMLR, Bejing, China, Proceedings of Machine Learning Research, vol 32, pp 1278–1286, URL http://proceedings.mlr.press/v32/rezende14.html

Rood T, van Gerven M, Lanillos P (2020) A deep active inference model of the rubber-hand illusion. In: Verbelen T, Lanillos P, Buckley CL, De Boom C (eds) Active Inference, Springer International Publishing, Cham, pp 84–91

Sancaktar C, van Gerven M, Lanillos P (2020) End-to-end pixel-based deep active inference for body perception and action. URL https://arxiv.org/abs/2001.05847

Schneider T (N.D.) Active inference for robotic manipulation, unpublished

Schneider T, Belousov B, Abdulsamad H, Peters J (2022) Active inference for robotic manipulation. arXiv preprint arXiv:220610313

Schulman J, Wolski F, Dhariwal P, Radford A, Klimov O (2017) Proximal policy optimization algorithms. URL https://arxiv.org/abs/1707.06347

Schwartenbeck P, Passecker J, Hauser TU, FitzGerald THB, Kronbichler M, Friston K (2018) Computational mechanisms of curiosity and goal-directed exploration. bioRxiv DOI 10.1101/411272, URL https://www.biorxiv.org/content/early/2018/09/07/411272, https://www.biorxiv.org/content/early/2018/09/07/411272.full.pdf

Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, Dieleman S, Grewe D, Nham J, Kalchbrenner N, Sutskever

I, Lillicrap TP, Leach M, Kavukcuoglu K, Graepel T, Hassabis D (2016) Mastering the game of go with deep neural networks and tree search. Nature 529(7587):484–489, DOI 10.1038/nature16961, URL https://doi.org/10.1038/nature16961

Sutton RS, Barto AG, et al (1998) Introduction to reinforcement learning. MIT press Cambridge

Ueltzhöffer K (2018) Deep active inference. Biol Cybern 112(6):547–573, DOI 10.1007/s00422-018-0785-7, URL https://doi.org/10.1007/s00422-018-0785-7

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

In this appendix, we tackle the question of how to estimate (28), 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:

$$Q(s_{t+1}|a_t) = \mathbb{E}_{Q_{\phi_s}(s_t)}\big[P_{\theta_s}(s_{t+1}|s_t, a_t)\big] \approx \frac{1}{N}\sum_{i=1}^{N} P_{\theta_s}(s_{t+1}|s_t = \hat{s}_t^i, a_t), \tag{34}$$

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, (34) 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[8]:

$$Q(s_{t+1}|a_t) \approx P_{\theta_s}(s_{t+1}|s_t = \hat{s}_t, a_t), \tag{35}$$

[8]Importantly, we chose $N = 1$ because for higher value of N, the estimator of $Q(s_{t+1}|a_t)$ would be a mixture of Gaussian with N components, effectively rendering the KL-divergence in Equation 40 analytically intractable.

where $\hat{s}_t^i$ 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 (2022) and define the prior preferences as:

$$P(o_\tau) = \frac{\exp(\psi r_\tau[o_\tau])}{\sum_{o_\tau} \exp(\psi r_\tau[o_\tau])}, \tag{36}$$

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:

$$\begin{aligned} \ln P(o_\tau) &= \psi r_\tau[o_\tau] - \ln \sum_{o_\tau} \exp(\psi r_\tau[o_\tau]) \\ &= \psi r_\tau[o_\tau] + C, \end{aligned} \tag{37}$$

where we used the fact that the summation over all o_τ is a normalization term, i.e., a constant. Using (37), we can now create an estimate of the extrinsic value as follows:

$$\mathbb{E}_{P_{\theta_o}(o_\tau|s_\tau)Q(s_\tau|a_t)}\big[\ln P(o_\tau)\big] \approx \frac{1}{M}\sum_{i=1}^{M} \ln P(o_\tau = \hat{o}_\tau^i) = \frac{1}{M}\sum_{i=1}^{M} \psi r_\tau[o_\tau^i] + C, \tag{38}$$

where $\hat{o}_\tau^i \sim P_{\theta_o}(o_\tau|s_\tau = \hat{s}_\tau^i)$ and $\hat{s}_\tau^i \sim Q(s_\tau|a_t)$. In what follows, we use $M = 1$ and discard the constant[9], which leads to a simplified version of the estimate:

$$\mathbb{E}_{P_{\theta_o}(o_\tau|s_\tau)Q(s_\tau|a_t)}\big[\ln P(o_\tau)\big] \triangleq \psi r_\tau[o_\tau] \triangleq \psi r_\tau, \tag{39}$$

[9]Removing a constant does not influence which policy is the best. Indeed, $\pi^* = \arg\max_\pi G(\pi) = \arg\max_\pi G(\pi) - C$. Importantly, removing the constant does not influence the latent representation as the gradients are not allowed to flow through the encoder, i.e., only the critic weights are updated when minimizing EFE.

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 $\tau = t+1$:

$$\begin{aligned} G_{t+1}(a_t) &\approx D_{\mathrm{KL}}\left[\, Q(s_{t+1}|a_t) || \, Q_{\phi_s}(s_{t+1})\right] - \mathbb{E}_{P_{\theta_o}(o_{t+1}|s_{t+1})Q(s_{t+1}|a_t)}\big[\ln P(o_{t+1})\big] \\ &\approx D_{\mathrm{KL}}\left[\, P_{\theta_s}(s_{t+1}|s_t = \hat{s}_t, a_t) || \, Q_{\phi_s}(s_{t+1})\right] - \psi r_{t+1}, \end{aligned} \tag{40}$$

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 6, the reward at time step $t+1$ is used to compute the target values that must be predicted by the critic network.

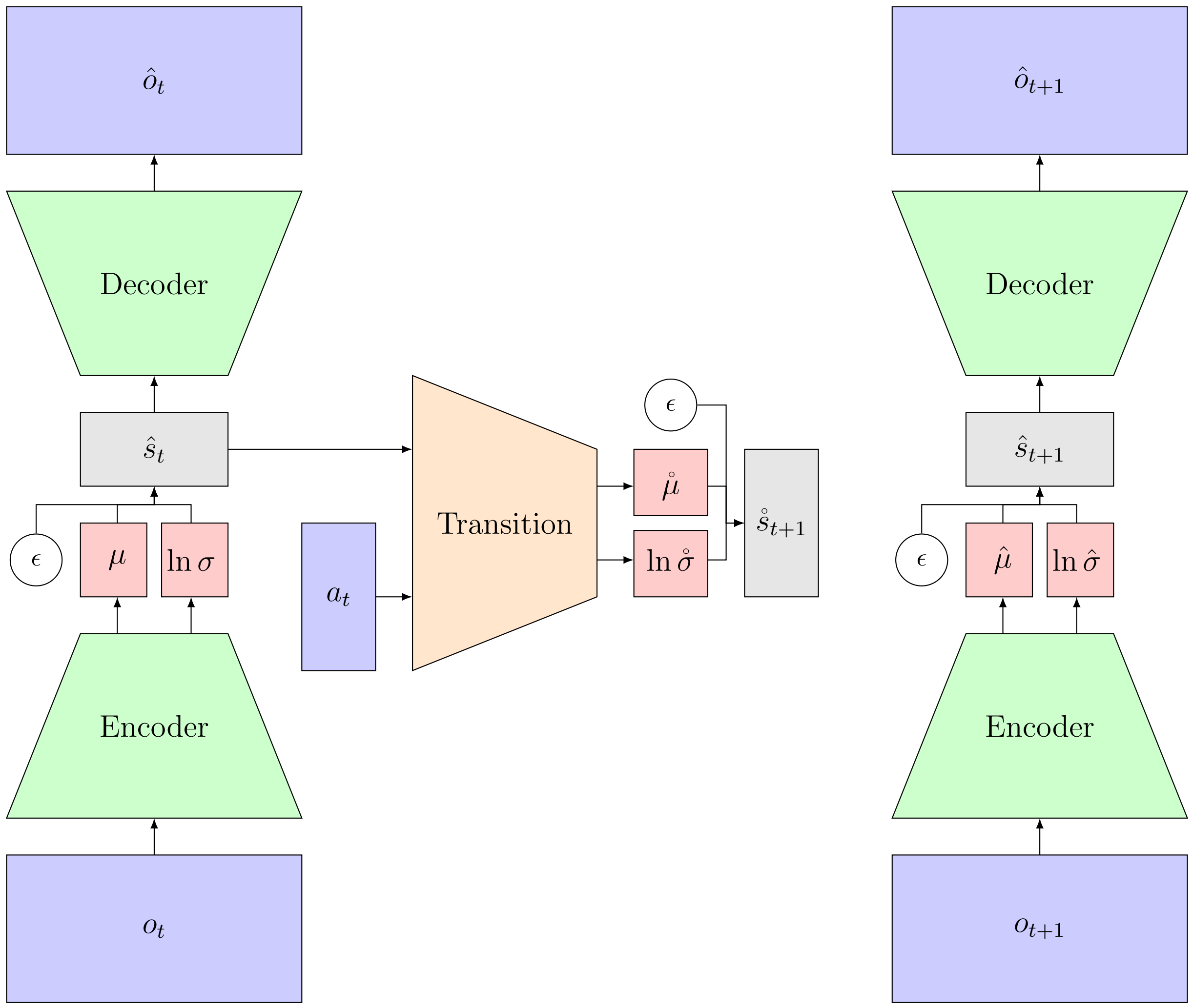


Figure 4: This figure illustrates the HMM agent. On the left and right, one can see two auto-encoders, i.e., one at time step t and one at time step $t + 1$. In the middle, the transition network takes as input the state and action at time t, i.e., $(\hat{s}_t, a_t)$, and outputs the mean $\mathring{\mu}$ and log variance $\ln \mathring{\sigma}$ of Gaussian distributions. By sampling the latent variable ϵ, and using the reparameterization 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.

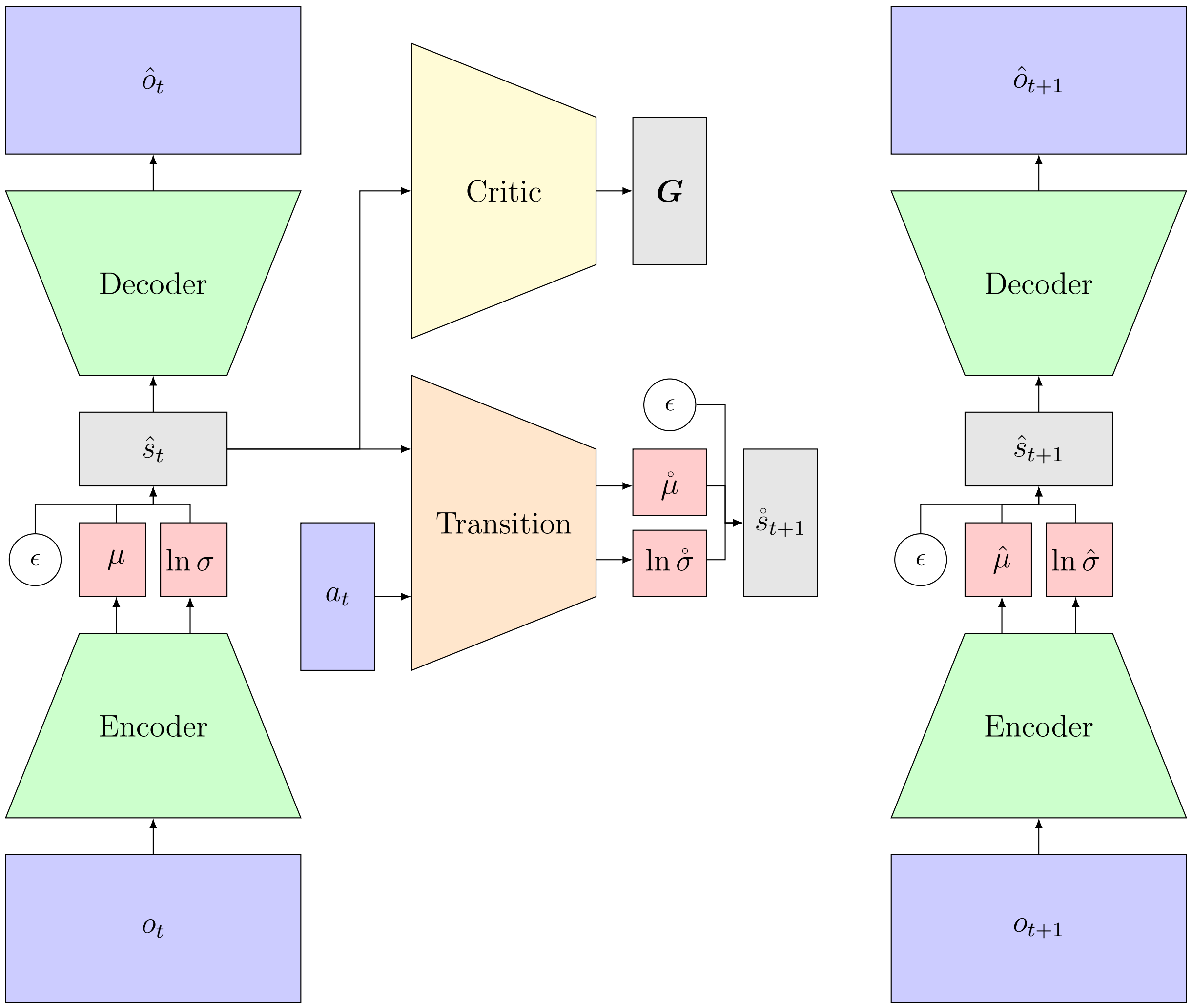


Figure 5: This figure illustrates the CHMM agent. The only new part is the critic network, which takes as input the hidden state at time t and ouputs the expected free energy of each action $\boldsymbol{G}$. Importantly, the CHMM takes actions according to the EFE.

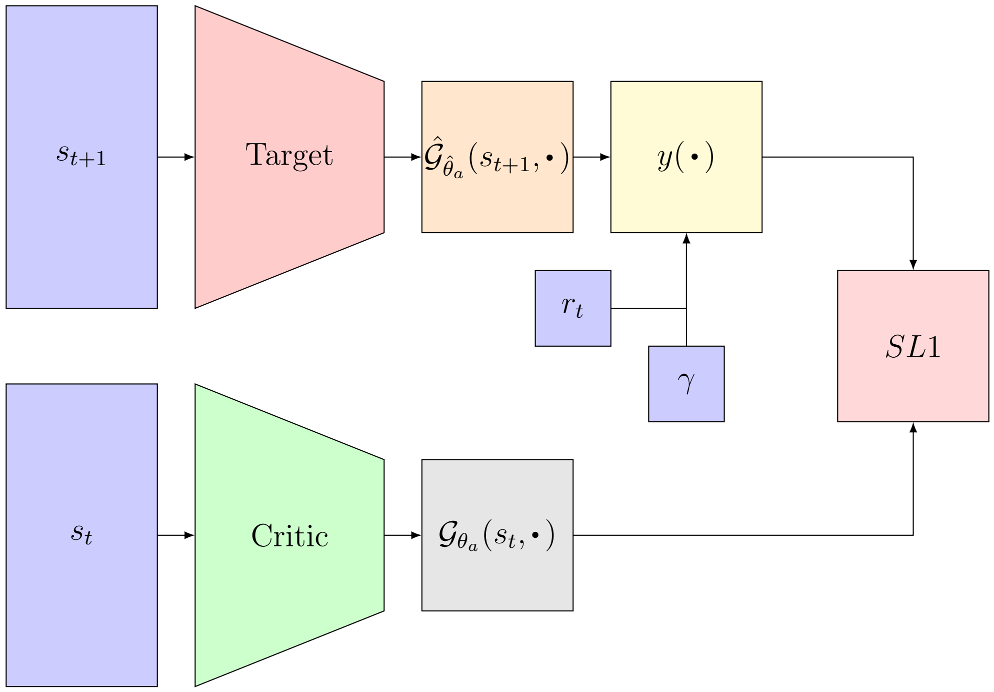
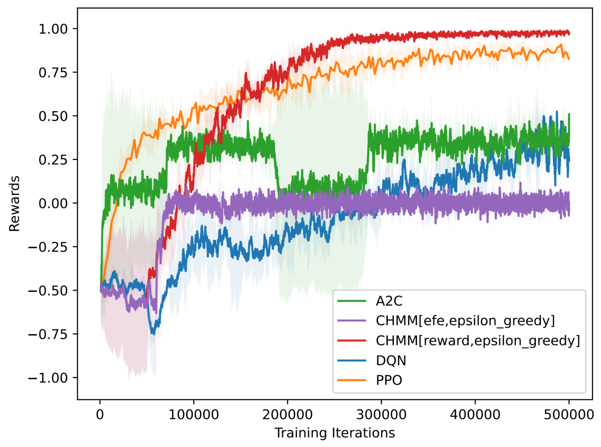
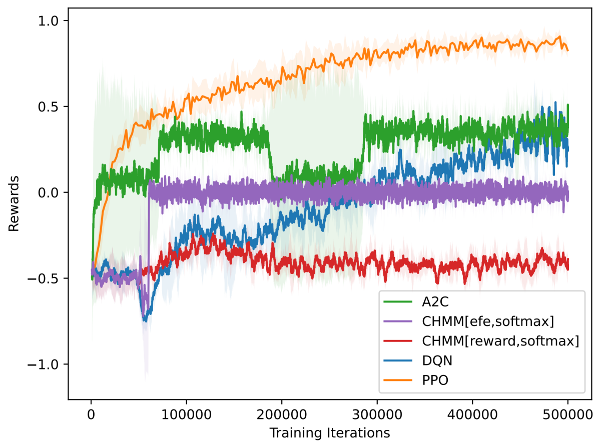

s_{t+1}
Target
$\hat{\mathcal{G}}_{\hat{\theta}_a}(s_{t+1}, \bullet)$
$y(\bullet)$
r_t
γ
$SL1$
s_t
Critic
$\mathcal{G}_{\theta_a}(s_t, \bullet)$

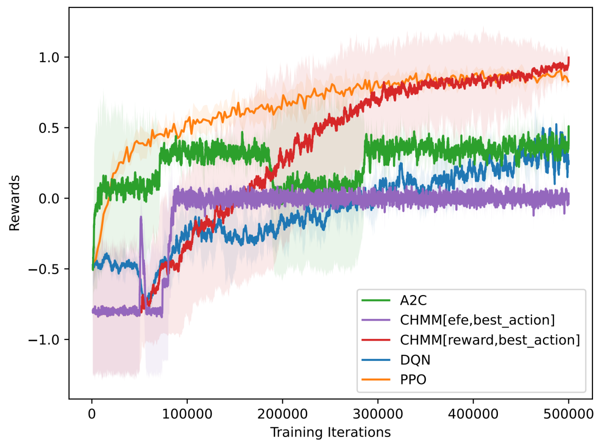
Figure 6: This figure illustrates the computation of the critic's loss function when the critic is only maximizing reward, i.e., when Equation 30 is used for the expected free energy. Briefly, the state s_t is fed into the Critic, and the state s_{t+1} is fed into the target network. The critic outputs 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 $y(\bullet)$. Finally, the goal is to minimize the SL1 between the prediction of the critic and the target values by changing the weights of the critic.



(a)

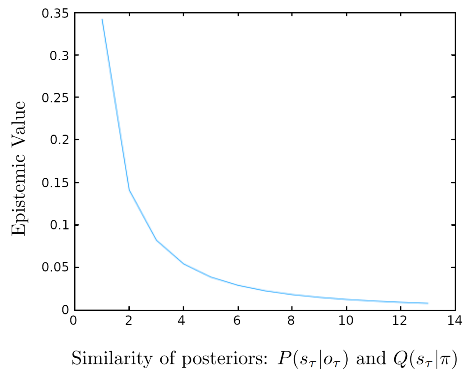


(b)

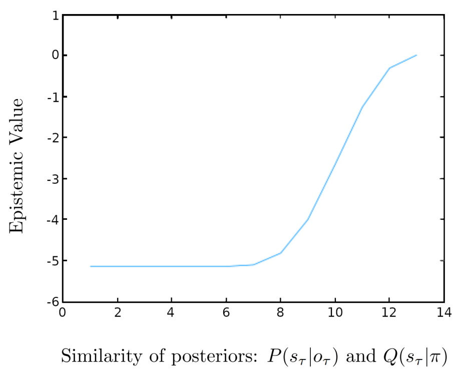


(c)

Figure 11: This figure illustrates the mean episodic reward gathered (during 500K training iterations) by the CHMM agents with (a) $\mathring{\epsilon}$-greedy, (b) softmax, and (c) best action selection. The CHMM performance is compared to the stable baseline implementation of PPO, A2C and DQN with default parameters.



(a)



(b)

Figure 14: The left-most figure shows the result of the first experiment where the likelihood was becoming more and more uniform. In this setting, the epistemic value encourages exploration, as maximizing the epistemic value makes the true posterior $P(s_\tau|o_\tau)$ as different as possible to the approximate posterior $Q(s_\tau|\pi)$, leftward on x-axis. However, the right-most figure illustrates the result of the second experiment where the prior over states was changing. In this case, the epistemic value does not promote exploration, as maximizing the epistemic value makes the true posterior $P(s_\tau|o_\tau)$ as similar as possible to the approximate posterior $Q(s_\tau|\pi)$, rightward on x-axis. Indeed, in this case, maximizing the epistemic value causes information to be lost.

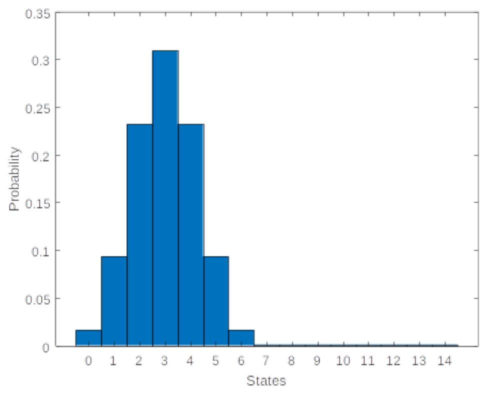
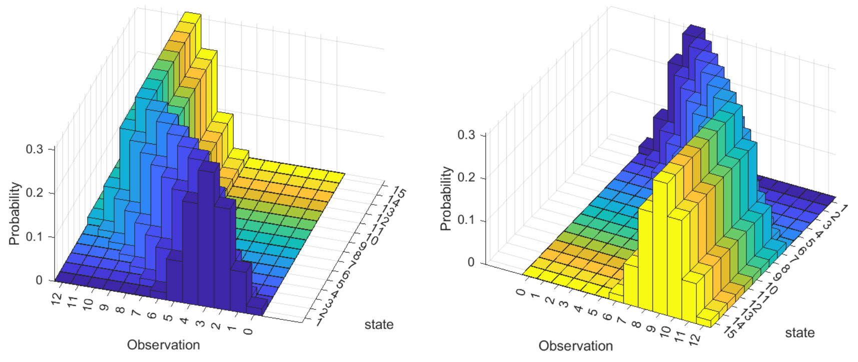
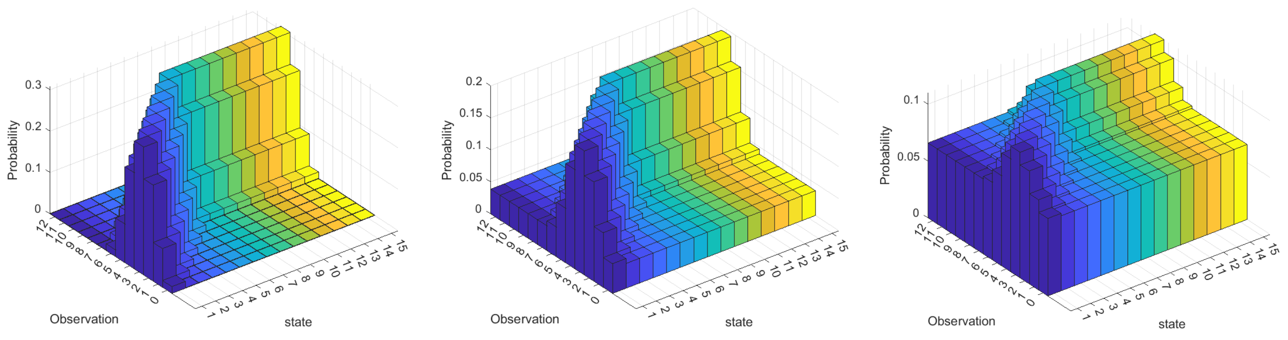


Figure 15: This figure illustrates the approximate posterior $Q(s_\tau|\pi)$, which is distributed according to a binomial distribution corresponding to 6 trials with a probability of success of 0.5.



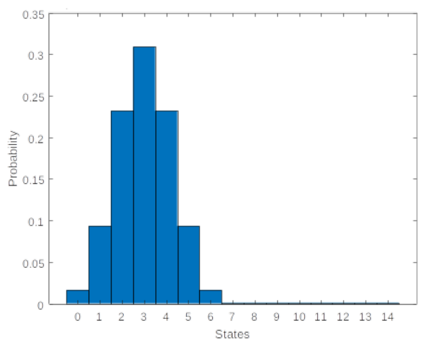
(a) (b)

Figure 16: This figure provides two views of the same likelihood mapping $P(o_\tau|s_\tau)$, i.e., one from the front and one from behind. The likelihood was created by sliding a binomial distribution (corresponding to 6 trials with a probability of success of 0.5) across the observation axis, each time the state increases by one. Finally, when the binomial reached its most extreme position, it stays the same for the remaining states.

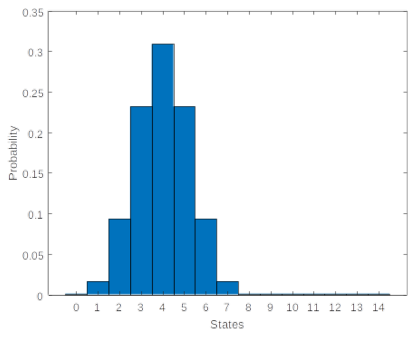


(a) (b) (c)

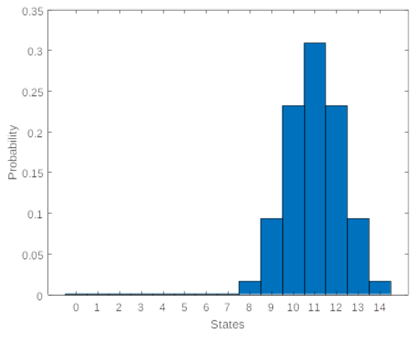
Figure 17: During the first experiment, we changed the likelihood mapping $P(o_\tau|s_\tau)$ by making it more and more uniform. This change is shown in the figure from left to right.



(a)

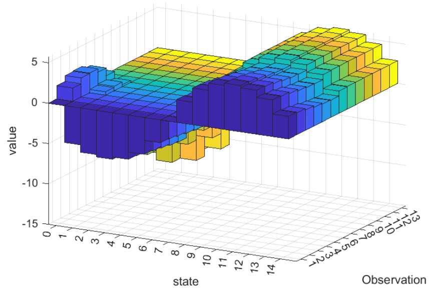


(b)

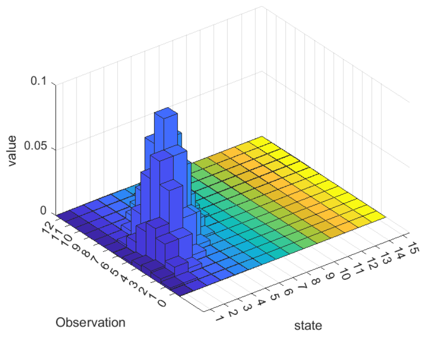


(c)

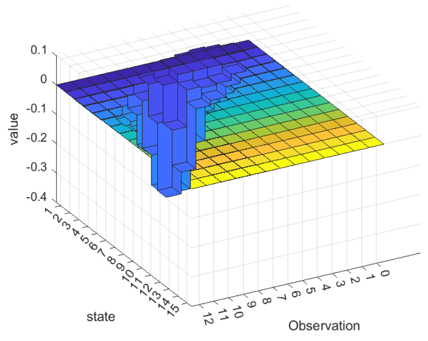
Figure 18: During the second experiment, we changed the prior over states $P(s_\tau)$ by making it more and more different to the approximate posterior $Q(s_\tau|\pi)$. This change is shown in the figure from left to right.



(a)



(b)



(c)

Figure 19: To compute the epistemic value, we first computed the difference between the logarithm of the true posterior $\ln P(s_\tau|o_\tau)$ and the logarithm of the approximate posterior $\ln Q(s_\tau|\pi)$ for all the values taken by the observation o_τ (c.f., left-most figure). Then, we computed the joint distribution $P(o_\tau|s_\tau)Q(s_\tau|\pi)$ used in the expectation (c.f., middle figure). Next, we compute the element-wise product between the matrices illustrated in the left-most and middle figures (c.f., right-most figure). Finally, the epistemic value is obtained by summing up all the elememts of the element-wise product. In this instance, the epistemic value will be strongly negative.