Deep active inference agents using Monte-Carlo methods

Zafeirios Fountas[*]
Emotech Labs &
WCHN, University College London
f@emotech.co

Noor Sajid
WCHN, University College London
noor.sajid.18@ucl.ac.uk

Pedro A.M. Mediano
e University of Cambridge
pam83@cam.ac.uk

Karl Friston
WCHN, University College London
k.friston@ucl.ac.uk

Abstract

Active inference is a Bayesian framework for understanding biological intelligence. The underlying theory brings together perception and action under one single imperative: minimizing free energy. However, despite its theoretical utility in explaining intelligence, computational implementations have been restricted to low-dimensional and idealized situations. In this paper, we present a neural architecture for building deep active inference agents operating in complex, continuous state-spaces using multiple forms of Monte-Carlo (MC) sampling. For this, we introduce a number of techniques, novel to active inference. These include: i) selecting free-energy-optimal policies via MC tree search, ii) approximating this optimal policy distribution via a feed-forward 'habitual' network, iii) predicting future parameter belief updates using MC dropouts and, finally, iv) optimizing state transition precision (a high-end form of attention). Our approach enables agents to learn environmental dynamics efficiently, while maintaining task performance, in relation to reward-based counterparts. We illustrate this in a new toy environment, based on the dSprites data-set, and demonstrate that active inference agents automatically create disentangled representations that are apt for modeling state transitions. In a more complex Animal-AI environment, our agents (using the same neural architecture) are able to simulate future state transitions and actions (i.e., plan), to evince reward-directed navigation - despite temporary suspension of visual input. These results show that deep active inference – equipped with MC methods – provides a flexible framework to develop biologically-inspired intelligent agents, with applications in both machine learning and cognitive science.

1 Introduction

A common goal in cognitive science and artificial intelligence is to emulate biological intelligence, to gain new insights into the brain and build more capable machines. A widely-studied neuroscience proposition for this is the free-energy principle, which views the brain as a device performing variational (Bayesian) inference [1, 2]. Specifically, this principle provides a framework for understanding biological intelligence, termed active inference, by bringing together perception and action under a single objective: minimizing free energy across time [3–7]. However, despite the potential of active inference for modeling intelligent behavior, computational implementations have been largely restricted to low-dimensional, discrete state-space tasks [8–11].

[*]Corresponding author

Preprint. Under review.

Recent advances have seen deep active inference agents solve more complex, continuous state-space tasks, including Doom [12], the mountain car problem [13–15], and several tasks based on the MuJoCo environment [16], many of which use amortization to scale-up active inference [13–15, 17]. A common limitation of these applications is a deviation from vanilla active inference in their ability to plan. For instance, Millidge [17] introduced an approximation of the agent's *expected free energy* (EFE), the quantity that drives action selection, based on bootstrap samples, while Tschantz *et al.* [16] employed a reduced version of EFE. Additionally, since all current approaches tackle low-dimensional problems, it is unclear how they would scale up to more complex domains. Here, we propose an extension of previous formulations that is closely aligned with active inference [4, 9] by estimating all EFE summands using a single deep neural architecture.

Our implementation of deep active inference focuses on ensuring both scalability and biological plausibility. We accomplish this by introducing MC sampling – at several levels – into active inference. For planning, we propose the use of MC tree search (MCTS) for selecting a free-energy-optimal policy. This is consistent with planning strategies employed by biological agents and provides an efficient way to select actions. Next, we approximate the optimal policy distribution using a feed-forward 'habitual' network. This is inspired by biological habit formation, when acting in familiar environments that relieves the computational burden of planning in commonly-encountered situations. Additionally, for both biological consistency and reducing computational burden, we predict model parameter belief updates using MC-dropouts, a problem previously tackled with networks ensembles. Lastly, inspired by neuromodulatory mechanisms in biological agents, we introduce a top-down mechanism that modulates precision over state transitions, which enhances learning of latent representations.

In what follows, we briefly review active inference. This is followed by a description of our deep active inference agent. We then evaluate the performance of this agent. Finally, we discuss the potential implications of this work.

2 Active Inference

Agents defined under active inference: $A)$ sample their environment and calibrate their internal generative model to best explain sensory observations (i.e., reduce surprise) and $B)$ perform actions under the objective of reducing their uncertainty about the environment. A more formal definition requires a set of random variables: s_t to represent the hidden state of the world at time t, o_t as the corresponding observation, $\pi = \{a_1, a_2, ..., a_T\}$ as a sequence of actions (typically referred to as 'policy' in the active inference literature) up to a given time horizon $T \in \mathbb{N}^+$, and $P(o_t, s_t; \theta)$ as the agent's generative model parameterized by θ. From this, the agent's surprise at time t can be defined as the negative log-likelihood $-\log P(o_t; \theta)$.

To address objective $A)$ under this formulation, the surprise of current observations can be indirectly minimized by optimizing the parameters, θ, using as a loss function the tractable expression:

$$-\log P_\theta(o_t; \theta) \leq \mathbb{E}_{Q(s_t)}\left[\log Q(s_t) - \log P(o_t, s_t; \theta)\right], \tag{1}$$

where $Q(s)$ is an arbitrary distribution of s. The RHS expression of this inequality is the *variational free energy* at time t. This quantity is commonly referred to as negative *evidence lower bound* [18] in variational inference. Furthermore, to realize objective $B)$, the expected surprise of future observations $-\log P(o_\tau|\theta)$ where $\tau \geq t -$ can be minimized by selecting the policy that is associated with the lowest EFE, G [19]:

$$G(\pi, \tau) = \mathbb{E}_{P(o_\tau|s_\tau,\theta)}\mathbb{E}_{Q(s_\tau,\theta|\pi)}\left[\log Q(s_\tau, \theta|\pi) - \log P(o_\tau, s_\tau, \theta|\pi)\right]. \tag{2}$$

Finally, the process of action selection in active inference is realized as sampling from the distribution

$$P(\pi) = \sigma\big(-\gamma G(\pi)\big) = \sigma\Big(-\gamma \sum_{\tau>t} G(\pi, \tau)\Big), \tag{3}$$

where γ is a temperature parameter and $\sigma(\cdot)$ the standard softmax function.

3 Deep Active Inference Architecture

In this section, we introduce a deep active inference model using neural networks, based on amortization and MC sampling.

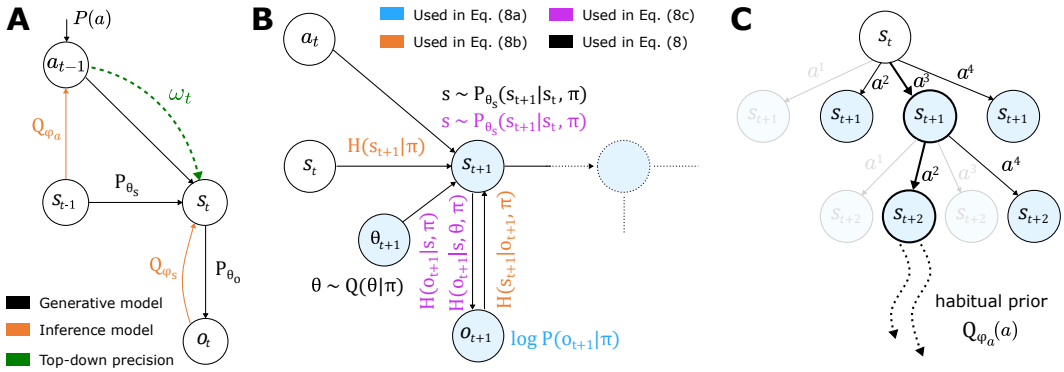

Figure 1: **A**: Schematic of model architecture and networks used during the learning process. Black arrows represent the generative model (P), orange arrows the recognition model (Q), and the green arrow the top-down attention (ω). **B**: Relevant quantities for the calculation of EFE G, computed by simulating the future using the generative model and ancestral sampling. Where appropriate, expectations are taken with a single MC sample. **C**: MCTS scheme used for planning and acting, using the habitual network to selectively explore new tree branches.

Throughout this section, we denote the parameters of the generative and recognition densities with θ and ϕ, respectively. The parameters are partitioned as follows: $\theta = \{\theta_o, \theta_s\}$, where θ_o parameterizes the observation function $P(o_t|s_t;\theta_o)$, and θ_s parameterizes the transition function $P(s_\tau|s_t, a_t;\theta_s)$. For the recognition density, $\phi = \{\phi_s, \phi_a\}$, where ϕ_s is the amortization parameters of the approximate posterior $Q_{\phi_s}(s_t)$ (i.e., the state encoder), and ϕ_a the amortization parameters of the approximate posterior $Q_{\phi_a}(a)$ (i.e., our habitual network).

3.1 Calculating variational and expected free energy

First, we extend the probabilistic graphical model (as defined in Sec. 2) to include the action sequences π and factorize the model based on Fig. 1A. We then exploit standard variational inference machinery to calculate the free energy for each time-step t as:

$$\begin{aligned} F_t = &- \mathbb{E}_{Q(s_t)}\left[\log P(o_t|s_t;\theta_o)\right] + D_{\mathrm{KL}}\left[Q_{\phi_s}(s_t) \parallel P(s_t|s_{t-1}, a_{t-1};\theta_s)\right] \\ &+ \mathbb{E}_{Q(s_t)}\left[D_{\mathrm{KL}}\left[Q_{\phi_a}(a_t) \parallel P(a_t)\right]\right] , \end{aligned} \tag{4}$$

where

$$P(a) = \sum_{\pi:a_1=a} P(\pi) \tag{5}$$

is the summed probability of all policies that begin with action a. We assume that s_t is normally distributed and o_t is Bernoulli distributed, with all parameters given by a neural network, parameterized by θ_o, θ_s, and ϕ_s for the observation, transition, and encoder models, respectively (see Sec. 3.2 for details about Q_{ϕ_a}). With this assumption, all the terms here are standard log-likelihood and KL terms easy to compute for Gaussian and Bernoulli distributions. The expectations over $Q_{\phi_s}(s_t)$ are taken via MC sampling, using a single sample from the encoder.

Next, we consider EFE. At time-step t and for a time horizon up to time T, EFE is defined as [4]:

$$G(\pi) = \sum_{\tau=t}^{T} G(\pi,\tau) = \sum_{\tau=t}^{T} \mathbb{E}_{\tilde{Q}}\left[\log Q(s_\tau,\theta|\pi) - \log \tilde{P}(o_\tau, s_\tau, \theta|\pi)\right] , \tag{6}$$

where $\tilde{Q} = Q(o_\tau, s_\tau, \theta|\pi)$ and $\tilde{P}(o_\tau, s_\tau, \theta|\pi) = P(o_\tau)Q(s_\tau|o_\tau)P(\theta|s_\tau, o_\tau)$. Following Schwartenbeck *et al.* [20], the EFE of a single time instance τ can be further decomposed as

$$G(\pi,\tau) = - \mathbb{E}_{\tilde{Q}}\left[\log P(o_\tau|\pi)\right] \tag{7a}$$

$$+ \mathbb{E}_{\tilde{Q}}\left[\log Q(s_\tau|\pi) - \log P(s_\tau|o_\tau,\pi)\right] \tag{7b}$$

$$+ \mathbb{E}_{\tilde{Q}}\left[\log Q(\theta|s_\tau,\pi) - \log P(\theta|s_\tau,o_\tau,\pi)\right] . \tag{7c}$$

Interestingly, each term constitutes a conceptually meaningful expression. The term (7a) corresponds to the likelihood assigned to the desired observations o_τ, and plays an analogous role to the notion of reward in the reinforcement learning literature [21]. The term (7b) corresponds to the mutual information between the agent's beliefs about its latent representation of the world, before and after making a new observation, and hence, it reflects a motivation to explore areas of the environment that resolve state uncertainty. Similarly, the term (7c) describes the tendency of active inference agents to reduce their uncertainty about model parameters via new observations and is usually referred to in the literature as active learning [3], novelty, or curiosity [20].

However, two of the three terms that constitute EFE cannot be easily computed as written in Eq. (7). To make computation practical, we will re-arrange these expressions and make further use of MC sampling to render these expressions tractable and re-write Eq. (7) as

$$G(\pi,\tau) = -\,\mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)Q(o_\tau|s_\tau,\theta\pi)}\left[\log P(o_\tau|\pi)\right] \tag{8a}$$

$$+\,\mathbb{E}_{Q(\theta|\pi)}\left[\mathbb{E}_{Q(o_\tau|\theta,\pi)}\,H(s_\tau|o_\tau,\pi) - H(s_\tau|\pi)\right] \tag{8b}$$

$$+\,\mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)}\,H(o_\tau|s_\tau,\theta,\pi) - \mathbb{E}_{Q(s_\tau|\pi)}\,H(o_\tau|s_\tau,\pi)\;, \tag{8c}$$

where these expressions can be calculated from the deep neural network illustrated in Fig. 1B. The derivation of Eq. (8) can be found in the supplementary material. To calculate the terms (8a) and (8b), we sample θ, s_τ and o_τ sequentially (through ancestral sampling) and then o_τ is compared with the prior distribution $\log P(o_\tau|\pi) = \log P(o_\tau)$. The parameters of the neural network θ are sampled from $Q(\theta)$ using the MC dropout technique [22]. Similarly, to calculate the expectation of $H(o_\tau|s_\tau,\pi)$, the same drawn θ is used again and s_τ is re-sampled for N times while, for $H(o_\tau|s_\tau,\theta,\pi)$, the set of parameters θ is also re-sampled N times. Finally, all entropies can be computed using the standard formulas for multivariate Gaussian and Bernoulli distributions.

3.2 Action selection and the habitual network

In active inference, agents choose an action given by their EFE. In particular, any given action is selected with a probability proportional to the accumulated negative EFE of the corresponding policies $G(\pi)$ (see Eq. (3) and Ref. [19]). However, computing G across all policies is costly since it involves making an exponentially-increasing number of predictions for T-steps into the future, and computing all the terms in Eq. (8). To solve this problem, we employ two methods operating in tandem. First, we employ standard MCTS [23–25], a search algorithm in which different potential future trajectories of states are explored in the form of a search tree (Fig. 1C), giving emphasis to the most likely future trajectories. This algorithm is used to calculate the distribution over actions $P(a)$, defined in Eq. (5), and control the agent's final decisions. Second, we make use of amortized inference through a habitual neural network that directly approximates the distribution over actions, which we parameterize by ϕ_a and denote $Q_{\phi_a}(a)$. In essence, $Q_{\phi_a}(a)$ acts as a variational posterior that approximates $P(a|s_t)$, with a prior $P(a)$, calculated by MCTS (see Fig. 1A). During learning, this network is trained to reproduce the last executed action a_{t-1} (selected by sampling $P(a)$) using the last state s_{t-1}. Since both tasks used in this paper (Sec. 4) have discrete action spaces $\mathcal{A}$, we define $Q_{\phi_a}(a)$ as a neural network with parameters ϕ_a and $|\mathcal{A}|$ softmax output units.

During the MCTS process, the agent generates a weighted search tree iteratively that is later sampled during action selection. In each single MCTS loop, one plausible state-action trajectory $(s_t, a_t, s_{t+1}, a_{t+1}, ..., s_\tau, a_\tau)$ – starting from the present time-step t – is calculated. For states that are explored for the first time, the distribution $P(s_{t+1}|s_t, a_t; \theta_s)$ is used. States that have been explored are stored in the *buffer* search tree and accessed during later loops of the same planning process. The weights of the search tree $\tilde{G}(a,s)$ represent the agent's best estimation for EFE after taking action a from state s. An upper confidence bound for $G(a,s)$ is defined as

$$U(s,a) = \tilde{G}(s,a) + c_{\text{explore}} \cdot Q_{\phi_a}(a|s) \cdot \frac{1}{1+N(a,s)}\;, \tag{9}$$

where $N(a,s)$ is the number of times that a was explored from state s, and c_{explore} a hyper-parameter that controls exploration. In each round, the EFE of the newly-explored parts of the trajectory is calculated and back-propagated to all visited nodes of the search tree. Additionally, actions are sampled in two ways. Actions from states that have been explored are sampled from $\sigma(U(a,s_t))$ while actions from new states are sampled from $Q_{\phi_a}(a)$.

Finally, the actions a_i that assemble the selected policy are drawn from $P(a) = \frac{N(a_i,s)}{\sum_j N(a_j,s)}$. In our implementation, the planning loop stops if either the process has identified a clear option (i.e. if $\max P(a) - 1/|\mathcal{A}| > T_{dec}$) or the maximum number of allowed loops has been reached.

Through the combination of the approximation $Q_{\phi_a}(a)$ and the MCTS, our agent has at its disposal two methods of action selection. We refer to $Q_{\phi_a}(a)$ as the *habitual* network, as it corresponds to a form of fast decision-making, quickly evaluating and selecting a action; in contrast with the more *deliberative* system that includes future imagination via MC tree traversals [26].

3.3 State precision and top-down attention

One of the key elements of our framework is the state transition model $P(s_t|s_{t-1}, a_{t-1}; \theta_s)$, that belongs to the agent's generative model. In our implementation, we take $s_t \sim \mathcal{N}(\mu, \sigma^2/\boldsymbol{\omega})$, where μ and σ come from the linear and softplus units (respectively) of a neural network with parameters θ_s applied to s_{t-1}, and, importantly, $\boldsymbol{\omega}$ is a *precision factor* (c.f. Fig. 1A) modulating the uncertainty on the agent's estimate of the hidden state of the environment [8]. We model the precision factor as a simple function of the belief update about the agent's current policy,

$$\boldsymbol{\omega} = \frac{a}{1 + e^{-\frac{D-b}{c}}} + d\,, \tag{10}$$

where $D = D_{\text{KL}}\big[Q_{\phi_a}(a) \parallel P(a)\big]$ and $\{a, b, c, d\}$ are fixed hyper-parameters. Note that $\boldsymbol{\omega}$ is a monotonically decreasing function of D, such that when the posterior belief about the current policy is similar to the prior, precision is high.

In cognitive terms, $\boldsymbol{\omega}$ can be thought of as a means of *top-down attention* [27], that regulates which transitions should be learnt in detail and which can be learnt less precisely. This attention mechanism acts as a form of resource allocation: if $D_{\text{KL}}\big[Q_{\phi_a}(a) \parallel P(a)\big]$ is high, then a habit has not yet been formed, reflecting a generic lack of knowledge. Therefore, the precision of the prior $P(s_t|s_{t-1}, a_{t-1}; \theta_s)$ (i.e., the belief about the current state before a new observation o_t has been received) is low, and less effort is spent learning $Q_{\phi_s}(s_t)$.

In practice, the effect of $\boldsymbol{\omega}$ is to *incentivize disentanglement* in the latent state representation s_t – the precision factor $\boldsymbol{\omega}$ is somewhat analogous to the β parameter in β-VAE [28], effectively pushing the state encoder $Q_{\phi_s}(s_t)$ to have independent dimensions (since $P(s_t|s_{t-1}, a_{t-1}; \theta_s)$ has a diagonal covariance matrix). As training progresses and the habitual network becomes a better approximation of $P(a)$, $\boldsymbol{\omega}$ is gradually increased, implementing a natural form of precision annealing.

4 Results

First, we present the two environments that were used to validate our agent's performance.

Dynamic dSprites We defined a simple 2D environment based on the dSprites dataset [29, 28]. This was used to i) quantify the agent's behavior against ground truth state-spaces and ii) evaluate the agent's ability to disentangle state representations. This is feasible as the dSprites data is designed for characterizing disentanglement, using a set of interpretable, independent ground-truth latent factors. In this task, which we call *object sorting*, the agent controls the position of the object via 4 different actions (right, left, up or down) and is required to sort single objects based on their shape (a latent factor). The agent receives reward when it moves the object across the bottom border, and the reward value depends on the shape and location as depicted in Fig. 2A. For the results presented in Section 4, the agent was trained in an on-policy fashion, with a batch size of 100.

Animal-AI We used a variation of *'preferences'* task from the Animal-AI environment [30]. The complexity of this, partially observable, 3D environment is the ideal test-bed for showcasing the agent's reward-directed exploration of the environment, whilst avoiding negative reward or getting stuck in corners. In addition, to test the agent's ability to rely on its internal model, we used a *lights-off* variant of this task, with temporary suspension of visual input at any given time-step with probability R. For the results presented in Section 4, the agent was trained in an off-policy fashion due to computational constraints. The training data for this was created using a simple rule: move in the direction of the greenest pixels.

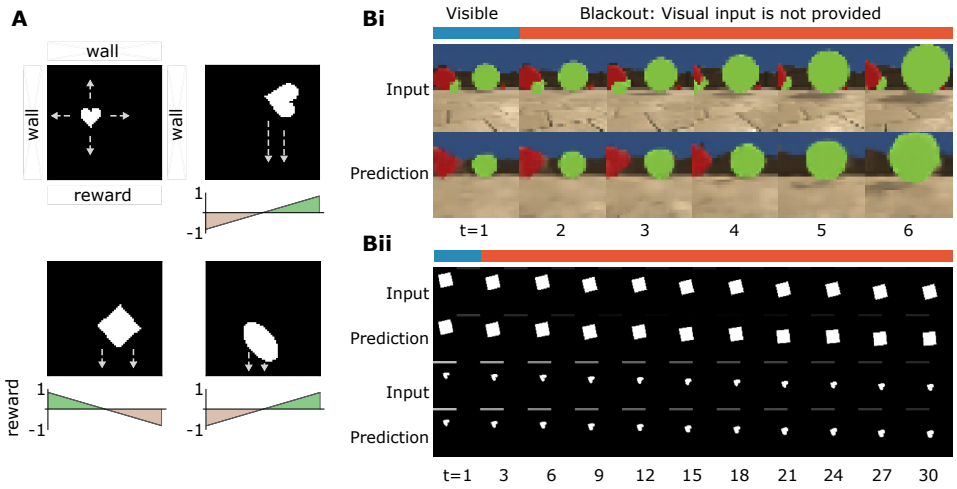

Figure 2: **A:** The proposed *object sorting* task based on the dSprites dataset. The agent can perform 4 actions; changing the position of the object in both axis. Reward is received if an object crosses the bottom boarder and differs for the 3 object shapes. **B:** Prediction of the visual observations under motion if input is hidden in both (**i**) AnimalAI and (**ii**) dynamic dSprites environments.

In the experiments that follow, we encode the actual reward from both environments as the prior distribution of future expected observations $\log P(o_\tau)$ or, in active inference terms, the expected outcomes. We optimized the networks using ADAM [31], with loss given in Eq. (4) and an extra regularization term $D_{\text{KL}}\big[Q_{\phi_s}(s_t) \parallel N(0,1)\big]$. The explicit training procedure is detailed in the supplementary material. The complete source-code, data, and pre-trained agents, is available on GitHub (https://github.com/zfountas/deep-active-inference-mc).

4.1 Learning environment dynamics and task performance

We initially show – through a simple visual demonstration (Fig. 2B) – that agents learn the environment dynamics with or without consistent visual input for both dynamic dSprites and AnimalAI. This is further investigated, for the dynamic dSprites, by evaluating task performance (Fig. 3A-C), as well as reconstruction loss for both predicted visual input and reward (Fig. 3D-E) during training.

To explore the effect of using different EFE functionals on behavior, we trained and compared active inference agents under three different formulations, all of which used the implicit reward function $\log P(o_\tau)$, against a baseline reward-maximizing agent. These include i) beliefs about the latent states (i.e., terms a,b from Eq. 7), ii) beliefs about both the latent states and model parameters (i.e., complete Eq. 7) and iii) beliefs about the latent states, with a down-weighted reward signal. We found that, although all agents exhibit similar performance in collecting rewards (Fig. 3B), active inference agents have a clear tendency to explore the environment (Fig. 3C). Interestingly, our results also demonstrate that all three formulations are better at reconstructing the expected reward, in comparison to a reward-maximizing baseline (Fig. 3D). Additionally, our agents are capable of reconstructing the current observation, as well as predicting 5 time-steps into the future, for all formulations of EFE, with similar loss with the baseline (Fig. 3E).

4.2 Disentanglement and transition learning

Disentanglement of latent spaces leads to lower dimensional temporal dynamics that are easier to predict [32]. Thus, generating a disentangled latent space s can be beneficial for learning the parameters of the transition function $P(s_\tau|s_t, a_t; \theta_s)$. Due to the similarity between the precision term $\boldsymbol{\omega}$ and the hyper-parameter β in β-VAE [28] discussed in Sec. 3.3, we hypothesized that $\boldsymbol{\omega}$ could play an important role in regulating transition learning. To explore this hypothesis, we compared the total correlation (as a metric for disentanglement [33]) of latent state beliefs between i) agents that have been trained with the different EFE functionals, ii) the baseline (reward-maximizing) agent, iii) an agent trained without top-down attention (although the average value of $\boldsymbol{\omega}$ was maintained), as

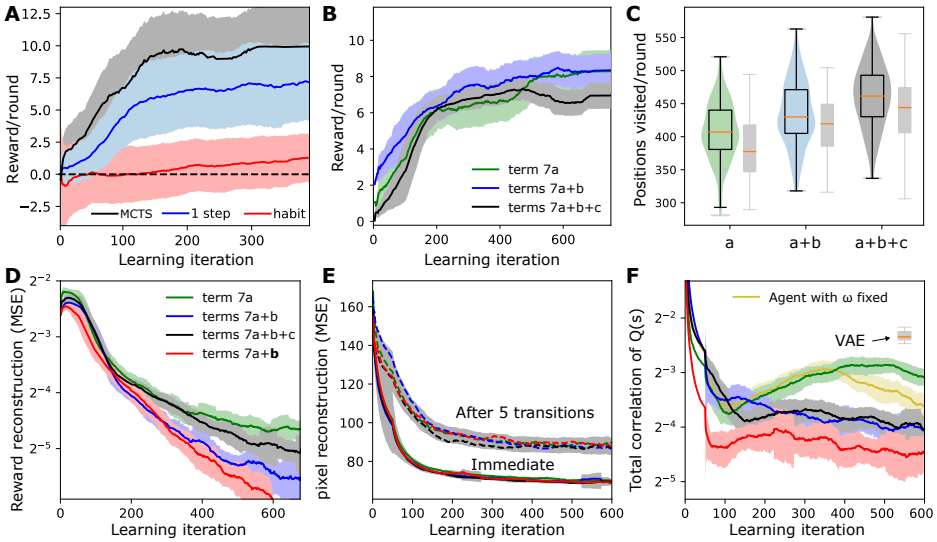

Figure 3: Agent's performance during on-policy training in the *object sorting* task. **A:** Comparison of different action selection strategies for the agent driven by the full Eq. (8). **B-C:** Comparison of agents driven by different functionals, limited to state estimations of a single step into the future. In C, the violin plots represent behavior driven by $P(a)$ (the planner) and the gray box plots driven by the habitual network $Q_{\phi_a}(a)$. **D-F:** Reconstruction loss and total correlation during learning for 4 different functionals.

well as iv) a simple variational autoencoder that received the same visual inputs. As seen in Fig. 3F, all active inference agents using $\boldsymbol{\omega}$ generated structures with significantly more disentanglement (see traversals in supp. material). Indeed, the performance ranking here is the same as in Fig. 3D, pointing to disentanglement as a possible reason for the performance difference in predicting rewards.

4.3 Navigation and planning in reward-based tasks

The training process in the dynamic dSprites environment revealed two types of behavior. Initially, we see epistemic exploration (i.e., curiosity), that is overtaken by reward seeking (i.e., goal-directed behavior) once the agent is reasonably confident about the environment. An example of this can be seen in the left trajectory plot in Fig. 4Ai, where the untrained agent – with no concept of reward – deliberates between multiple options and chooses the path that enables it to quickly move to the next round. The same agent, after 700 learning iterations, can now optimally plan where to move the current object, in order to maximize potential reward, $\log P(o_\tau)$. We next investigated the sensitivity when deciding, by changing the threshold T_{dec}. We see that changing the threshold has clear implications for the distribution of explored alternative trajectories i.e., number of simulated states (Fig. 4Aii). This plays an important role in the performance, with maximum performance found at $T_{dec} \approx 0.8$ (Fig. 4Aiii).

Agents trained in the Animal-AI environment also exhibit interesting (and intelligent) behavior. Here, the agent is able to make complex plans, by avoiding obstacles with negative reward and approaching expected outcomes (red and green objects respectively, Fig. 4Bi). Maximum performance can be found for 16 MCTS loops and $T_{dec} \approx 0.1$ (Fig. 4Bii; details in the supplementary material). When deployed in 'lights-off' experiments, the agent can successfully maintain an accurate representation of the world state and simulate future plans despite temporary suspension of visual input (Fig. 2B). This is particularly interesting because $P(s_{t+1}|s_t, a_t; \theta_s)$ is defined as a feed-forward network, without the ability to maintain memory of states before t. As expected, the agent's ability to operate in this set-up becomes progressively worse the longer the visual input is removed, while shorter decision thresholds are found to preserve performance longer (Fig. 4Biii).

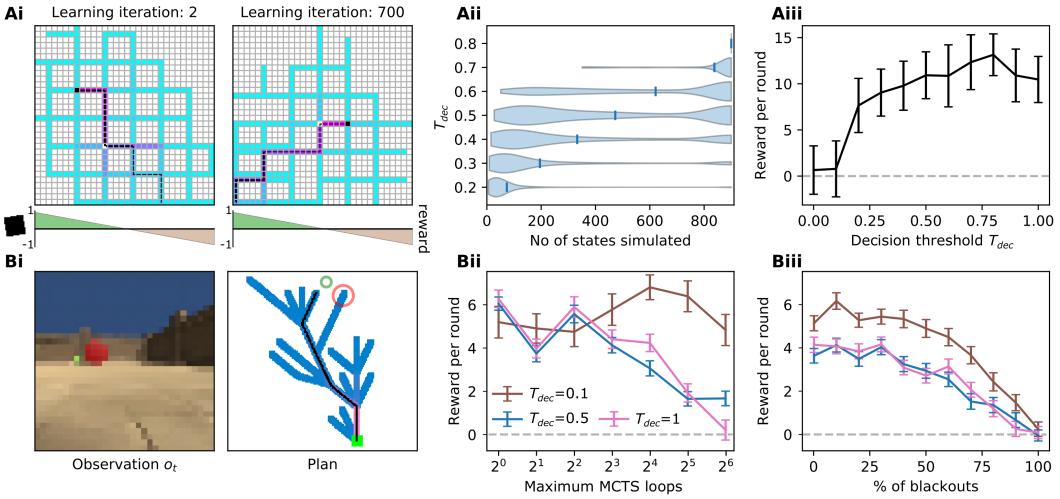

Figure 4: Agent's planning performance. **A:** Dynamic dSprites. **i)** Example planned trajectory plots with number of visits per state (blue-pink color map) and the selected policy (black lines). **ii)** The effect of decision threshold T_{dec} on the number of simulated states and **iii)** the agent's performance. **B:** Animal-AI. **i)** Same as in A. **ii)** System performance over hyper-parameters and **iii)** in the *lights-off* task. Error bars in Aiii denote standard deviation and in B standard error of the mean.

5 Concluding Remarks

The attractiveness of active inference inherits from the biological plausibility of the framework [4, 34, 35]. Accordingly, we focused on scaling-up active inference inspired by neurobiological structure and function that supports intelligence. This is reflected in the hierarchical generative model, where the higher-level policy network contextualizes lower-level state representations. This speaks to a separation of temporal scales afforded by cortical hierarchies in the brain and provides a flexible framework to develop biologically-inspired intelligent agents.

We introduced MCTS for tackling planning problems with vast search spaces [23, 36, 24, 37, 38]. This approach builds upon Çatal *et al.*'s [39] deep active inference proposal, to use tree search to recursively re-evaluate EFE for each policy, but is computationally more efficient. Additionally, using MCTS offers an *Occam's window* for policy pruning; that is, we stop evaluating a policy path if its EFE becomes much higher than a particular upper confidence bound. This pruning drastically reduces the number of paths one has to evaluate. It is also consistent with biological planning, where agents adopt brute force exploration of possible paths in a decision tree, up to a resource-limited finite depth [40]. This could be due to imprecise evidence about different future trajectories [41] where environmental constraints subvert evaluation accuracy [42, 43] or alleviate computational load [44]. Previous work addressing the depth of possible future trajectories in human subjects under changing conditions shows that both increased cognitive load [43] and time constraints [45, 46, 42] reduce search depth. Huys *et al.* [44] highlighted that in tasks involving alleviated computational load, subjects might evaluate only subsets of decision trees. This is consistent with our experiments as the agent selects to evaluate only particular trajectories based on their prior probability to occur.

We have shown that the precision factor, $\boldsymbol{\omega}$, can be used to incorporate uncertainty over the prior and enhances disentanglement by encouraging statistical independence between features [47–50]. This is precisely why it has been associated with attention [51]; a signal that shapes uncertainty [52]. Attention enables flexible modulation of neural activity that allows behaviorally relevant sensory data to be processed more efficiently [53, 54, 27]. The neural realizations of this have been linked with neuromodulatory systems, e.g., cholinergic and noradrenergic [55–59]. In active inference, they have been associated specifically with noradrenaline for modulating uncertainty about state transitions [8], noradrenergic modulation of visual attention [60] and dopamine for policy selection [4, 60].

A limitation of this work lies in its comparison to reward-maximizing agents. That is, if the specific goal is to maximize reward, then it is not clear whether deep active inference (i.e., full specification of EFE) has any performance benefits over simple reward-seeking agents (i.e., using only Eq. 7a). We

emphasize, however, that the primary purpose of the active inference framework is to serve as a model for biological cognition, and not as an optimal solution for reward-based tasks. Therefore, we have deliberately not focused on bench-marking performance gains against state-of-the-art reinforcement learning agents, although we hypothesize that insights from active inference could prove useful in complex environments where either reward maximization isn't the objective, or in instances where direct reward maximization leads to sub-optimal performance.

There are several extensions that can be explored, such as testing whether performance would increase with more complex, larger neural networks, e.g., using LSTMs to model state transitions. One could also assess if including episodic memory would finesse EFE evaluation over a longer time horizon, without increasing computational complexity. Future work should also test how performance shifts if the objective of the task changes. Lastly, it might be neurobiologically interesting to see whether the generated disentangled latent structures are apt for understanding functional segregation in the brain.

6 Acknowledgements

The authors would like to thank Sultan Kenjeyev for his valuable contributions and comments on early versions of the model presented in the current manuscript and Emotech team for the great support throughout the project. NS was funded by the Medical Research Council (MR/S502522/1). PM and KJF were funded by the Wellcome Trust (Ref: 210920/Z/18/Z - PM; Ref: 088130/Z/09/Z - KJF).

References

[1] Karl J Friston. The free-energy principle: A unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138, 2010.

[2] Karl J Friston. A free energy principle for a particular physics. *arXiv preprint arXiv:1906.10184*, 2019.

[3] Karl J Friston, Thomas. FitzGerald, Francesco Rigoli, Philipp Schwartenbeck, John O'Doherty, and Giovanni Pezzulo. Active inference and learning. *Neuroscience & Biobehavioral Reviews*, 68:862–79, 2016.

[4] Karl J Friston, Thomas FitzGerald, Francesco Rigoli, Philipp Schwartenbeck, and Giovanni Pezzulo. Active inference: A process theory. *Neural Computation*, 29(1):1–49, 2017.

[5] Karl J Friston, Thomas Parr, and Bert de Vries. The graphical brain: Belief propagation and active inference. *Network Neuroscience*, 1(4):381–414, 2017.

[6] Giovanni Pezzulo, Francesco Rigoli, and Karl J Friston. Hierarchical active inference: A theory of motivated control. *Trends in Cognitive Sciences*, 22(4):294–306, 2018.

[7] Lancelot Da Costa, Thomas Parr, Noor Sajid, Sebastijan Veselic, Victorita Neacsu, and Karl Friston. Active inference on discrete state-spaces: A synthesis. *arXiv preprint arXiv:2001.07203*, 2020.

[8] Thomas Parr and Karl J Friston. Uncertainty, epistemics and active inference. *Journal of The Royal Society Interface*, 14(136):20170376, 2017.

[9] Karl J Friston, Richard Rosch, Thomas Parr, Cathy Price, and Howard Bowman. Deep temporal models and active inference. *Neuroscience and Biobehavioral Reviews*, 90:486—501, 2018.

[10] Noor Sajid, Philip J Ball, and Karl J Friston. Active inference: Demystified and compared. *arXiv preprint arXiv:1909.10863*, 2019.

[11] Casper Hesp, Ryan Smith, Micah Allen, Karl J Friston, and Maxwell Ramstead. Deeply felt affect: The emergence of valence in deep active inference. *PsyArXiv*, 2019.

[12] Maell Cullen, Ben Davey, Karl J Friston, and Rosalyn J. Moran. Active inference in OpenAI Gym: A paradigm for computational investigations into psychiatric illness. *Biological Psychiatry: Cognitive Neuroscience and Neuroimaging*, 3(9):809–818, 2018.

[13] Karl J Friston, Jean Daunizeau, and Stefan J Kiebel. Reinforcement learning or active inference? *PLoS ONE*, 4(7):e6421, 2009.

[14] Kai Ueltzhöffer. Deep active inference. *Biological Cybernetics*, 112(6):547–573, 2018.

[15] Ozan Çatal, Johannes Nauta, Tim Verbelen, Pieter Simoens, and Bart Dhoedt. Bayesian policy selection using active inference. *arXiv preprint arXiv:1904.08149*, 2019.

[16] Alexander Tschantz, Manuel Baltieri, Anil Seth, Christopher L Buckley, et al. Scaling active inference. *arXiv preprint arXiv:1911.10601*, 2019.

[17] Beren Millidge. Deep active inference as variational policy gradients. *Journal of Mathematical Psychology*, 96:102348, 2020.

[18] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[19] Thomas Parr and Karl J Friston. Generalised free energy and active inference. *Biological Cybernetics*, 113(5-6):495–513, 2019.

[20] Philipp Schwartenbeck, Johannes Passecker, Tobias U Hauser, Thomas HB FitzGerald, Martin Kronbichler, and Karl J Friston. Computational mechanisms of curiosity and goal-directed exploration. *eLife*, 8:e41703, 2019.

[21] Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

[22] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.

[23] Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *International Conference on Computers and Games*, pages 72–83. Springer, 2006.

[24] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012.

[25] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354–359, 2017.

[26] Matthijs Van Der Meer, Zeb Kurth-Nelson, and A David Redish. Information processing in decision-making systems. *The Neuroscientist*, 18(4):342–359, 2012.

[27] Anna Byers and John T. Serences. Exploring the relationship between perceptual learning and top-down attentional control. *Vision Research*, 74:30 – 39, 2012.

[28] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations*, 2 (5):6, 2017.

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

[30] Matthew Crosby, Benjamin Beyret, and Marta Halina. The Animal-AI olympics. *Nature Machine Intelligence*, 1(5):257–257, 2019.

[31] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[32] Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li F Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction. In *Advances in Neural Information Processing Systems*, pages 517–526, 2018.

[33] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.

[34] Takuya Isomura and Karl J Friston. In vitro neural networks minimise variational free energy. *Nature Scientific Reports*, 8(1):1–14, 2018.

[35] Rick A Adams, Stewart Shipp, and Karl J Friston. Predictions not commands: Active inference in the motor system. *Brain Structure and Function*, 218(3):611–643, 2013.

[36] Levente Kocsis and Csaba Szepesvári. Bandit based Monte-Carlo planning. In *European Conference on Machine Learning*, pages 282–293. Springer, 2006.

[37] Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard L Lewis, and Xiaoshi Wang. Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning. In *Advances in Neural Information Processing Systems*, pages 3338–3346, 2014.

[38] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

[39] Ozan Çatal, Tim Verbelen, Johannes Nauta, Cedric De Boom, and Bart Dhoedt. Learning perception and planning with deep active inference. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3952–3956, 2020.

[40] Joseph Snider, Dongpyo Lee, Howard Poizner, and Sergei Gepshtein. Prospective optimization with limited resources. *PLoS Computational Biology*, 11(9), 2015.

[41] Alec Solway and Matthew M Botvinick. Evidence integration in model-based tree search. *Proceedings of the National Academy of Sciences*, 112(37):11708–11713, 2015.

[42] Bas van Opheusden, Gianni Galbiati, Zahy Bnaya, Yunqi Li, and Wei Ji Ma. A computational model for decision tree search. In *CogSci.*, 2017.

[43] Dennis H Holding. Counting backward during chess move choice. *Bulletin of the Psychonomic Society*, 27(5):421–424, 1989.

[44] Quentin JM Huys, Neir Eshel, Elizabeth O'Nions, Luke Sheridan, Peter Dayan, and Jonathan P Roiser. Bonsai trees in your head: How the Pavlovian system sculpts goal-directed choices by pruning decision trees. *PLoS Computational Biology*, 8(3), 2012.

[45] Bruce D Burns. The effects of speed on skilled chess performance. *Psychological Science*, 15 (7):442–447, 2004.

[46] Frenk Van Harreveld, Eric-Jan Wagenmakers, and Han LJ Van Der Maas. The effects of time pressure on chess skill: An investigation into fast and slow processes underlying expert performance. *Psychological Research*, 71(5):591–597, 2007.

[47] Emile Mathieu, Tom Rainforth, N Siddharth, and Yee Whye Teh. Disentangling disentanglement in variational autoencoders. *arXiv preprint arXiv:1812.02833*, 2018.

[48] Minyoung Kim, Yuting Wang, Pritish Sahu, and Vladimir Pavlovic. Bayes-Factor-VAE: Hierarchical Bayesian deep auto-encoder models for factor disentanglement. In *IEEE International Conference on Computer Vision*, pages 2979–2987, 2019.

[49] Hadi Fatemi Shariatpanahi and Majid Nili Ahmadabadi. Biologically inspired framework for learning and abstract representation of attention control. In *Attention in Cognitive Systems. Theories and Systems from an Interdisciplinary Viewpoint*, pages 307–324, 2007.

[50] Alexander Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo J Rezende. Towards interpretable reinforcement learning using attention augmented agents. In *Advances in Neural Information Processing Systems*, pages 12329–12338, 2019.

[51] Thomas Parr, David A. Benrimoh, Peter Vincent, and Karl J Friston. Precision and false perceptual inference. *Frontiers in Integrative Neuroscience*, 12:39, 2018.

[52] Peter Dayan, Sham Kakade, and Read P Montague. Learning and selective attention. *Nature Neuroscience*, 3(11):1218–1223, 2000.

[53] Farhan Baluch and Laurent Itti. Mechanisms of top-down attention. *Trends in Neurosciences*, 34(4):210–224, 2011.

[54] Yuka Sasaki, Jose E Nanez, and Takeo Watanabe. Advances in visual perceptual learning and plasticity. *Nature Reviews Neuroscience*, 11(1):53–60, 2010.

[55] Michael I Posner and Steven E Petersen. The attention system of the human brain. *Annual Review of Neuroscience*, 13(1):25–42, 1990.

[56] Peter Dayan and Angela J Yu. ACh, uncertainty, and cortical inference. In *Advances in Neural Information Processing Systems*, pages 189–196. 2002.

[57] Q Gu. Neuromodulatory transmitter systems in the cortex and their role in cortical plasticity. *Neuroscience*, 111(4):815–835, 2002.

[58] Angela J Yu and Peter Dayan. Uncertainty, neuromodulation, and attention. *Neuron*, 46(4): 681–692, 2005.

[59] Rosalyn J Moran, Pablo Campo, Mkael Symmonds, Klaas E Stephan, Raymond J Dolan, and Karl J Friston. Free energy, precision and learning: The role of cholinergic neuromodulation. *Journal of Neuroscience*, 33(19):8227–8236, 2013.

[60] Thomas Parr. *The Computational Neurology of Active Vision*. PhD thesis, University College London, 2019.

7 Supplementary material

7.1 Expected free energy derivation

Here, we provide the steps needed to derive Eq. (8) from Eq. (7). The term (7b) can be re-written as:

$$\begin{aligned}
&\mathbb{E}_{\tilde{Q}}\left[\log Q(s_\tau|\pi) - \log Q(s_\tau|o_\tau,\pi)\right] = \\
&\quad = \mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)Q(o_\tau|s_\tau,\theta,\pi)}\left[\log Q(s_\tau|\pi) - \log Q(s_\tau|o_\tau,\pi)\right] \\
&\quad = \mathbb{E}_{Q(\theta|\pi)}\left[\mathbb{E}_{Q(s_\tau|\theta,\pi)}\log Q(s_\tau|\pi) - \mathbb{E}_{Q(s_\tau,o_\tau|\theta,\pi)}\log Q(s_\tau|o_\tau,\pi)\right] \\
&\quad = \mathbb{E}_{Q(\theta|\pi)}\left[\mathbb{E}_{Q(o_\tau|\theta,\pi)}\,H(s_\tau|o_\tau,\pi) - H(s_\tau|\pi)\right],
\end{aligned}$$

where we have only used the definition $\tilde{Q} = Q(o_\tau, s_\tau, \theta|\pi)$, and the definition of the standard (and conditional) Shannon entropy.

Next, the term (7c) can be re-written as:

$$\begin{aligned}
&\mathbb{E}_{\tilde{Q}}\left[\log Q(\theta|s_\tau,\pi) - \log Q(\theta|s_\tau,o_\tau,\pi)\right] = \\
&\qquad = \mathbb{E}_{Q(s_\tau,\theta,o_\tau|\pi)}\left[\log Q(o_\tau|s_\tau,\pi) - \log Q(o_\tau|s_\tau,\theta,\pi)\right] \\
&\qquad = \mathbb{E}_{Q(s_\tau|\pi)Q(o_\tau|s_\tau,\pi)}\log Q(o_\tau|s_\tau,\pi) \\
&\qquad\quad - \mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)Q(o_\tau|s_\tau,\theta\pi)}\log Q(o_\tau|s_\tau,\theta,\pi) \\
&\qquad = \mathbb{E}_{Q(\theta|\pi)Q(s_\tau|\theta,\pi)}\,H(o_\tau|s_\tau,\theta,\pi) - \mathbb{E}_{Q(s_\tau|\pi)}\,H(o_\tau|s_\tau,\pi),
\end{aligned}$$

where the first equality is obtained via a normal Bayes inversion, and the second via the factorization of $\tilde{Q}$.

7.2 Training Procedure

The model presented here was implemented in Python and the library TensorFlow 2.0. We initialized 3 different ADAM optimizers, which we used in parallel, to allow learning parameters with different rates. The networks Q_{ϕ_s}, P_{θ_o} were optimized using an initial learning rate of 10^{-3} and, as a loss function, the first two terms of Eq. (4). In experiments where regularization was used, the loss function used by this optimizer was adjusted to

$$\begin{aligned}
L_{\phi_s,\theta_o} = &-\mathbb{E}_{Q(s_t)}\left[\log P(o_t|s_t;\theta_o)\right] + \gamma D_{\mathrm{KL}}\big[Q_{\phi_s}(s_t) \parallel P(s_t|s_{t-1},a_{t-1};\theta_s)\big] \\
&+ (1-\gamma)D_{\mathrm{KL}}\big[Q_{\phi_s}(s_t) \parallel N(0,1)\big],
\end{aligned} \tag{11}$$

where γ is a hyper parameter, starting with value 0 and gradually increasing to 0.8. In our experiments, we found that the effect of regularization is only to improve the speed of convergence and not the behavior of the agent and, thus, it can be safely omitted.

The parameters of the network P_{θ_s} were optimized using a rate of 10^{-4} and only the second term of Eq. (4) as a loss. Finally, the parameters of Q_{ϕ_a} were optimized with a learning rate of 10^{-4} and only the final term of Eq. (4) as a loss. For all presented experiments and learning curves, batch size was set to 50. A learning iteration is defined as 1000 optimization steps with new data generated from the corresponding environment.

In order to learn to plan further into the future, the agents were trained to map transitions every 5 simulation time-steps in dynamic dSprites and 3 simulation time-steps in Animal-AI. Finally, the runtime of the results presented here is as follows. For the agents in the dynamic dSprites environment, training of the final version of the agents took approximately 26 hours per version (on-policy, 700 learning iterations) using an NVIDIA Titan RTX GPU. Producing the learning and performance curves in Fig. 3, took 10 hours per agent when the 1-step and habitual strategies were employed and approximately 4 days when the full MCTS planner was used (Fig. 3A). For the Animal-AI environment, off-policy training took approximately 9 hours per agent, on-policy training took 8 days and, the results presented in Fig. 4 took approximately 4 days, using an NVIDIA GeForce GTX 1660 super GPU (CPU: i7-4790k, RAM: 16GB DDR3).

7.3 Model parameters

In both simulated environments, the network structure used was almost identical, consisting of convolutional, deconvolutional, fully-connected and dropout layers (Fig. 5). In both cases, the dimensionality of the latent space s was 10. For the top-down attention mechanism, the parameters used were $a = 2, b = 0.5, c = 0.1$ and $d = 5$ for the Animal-AI environment and $a = 1, b = 25, c = 5$ and $d = 1.5$ for dynamic dSprites. The action space was $|\mathcal{A}| = 3$ for Animal-AI and $|\mathcal{A}| = 4$ for dynamic dSprites. Finally, with respect to the planner, we set $c_{\text{explore}} = 1$ in both cases, $T_{dec} = 0.8$ (when another value is not specifically mentioned), the depth of MCTS simulation rollouts was set to 3, while the maximum number of MCTS loops was set to 300 for dynamic dSprites and 100 for Animal-AI.

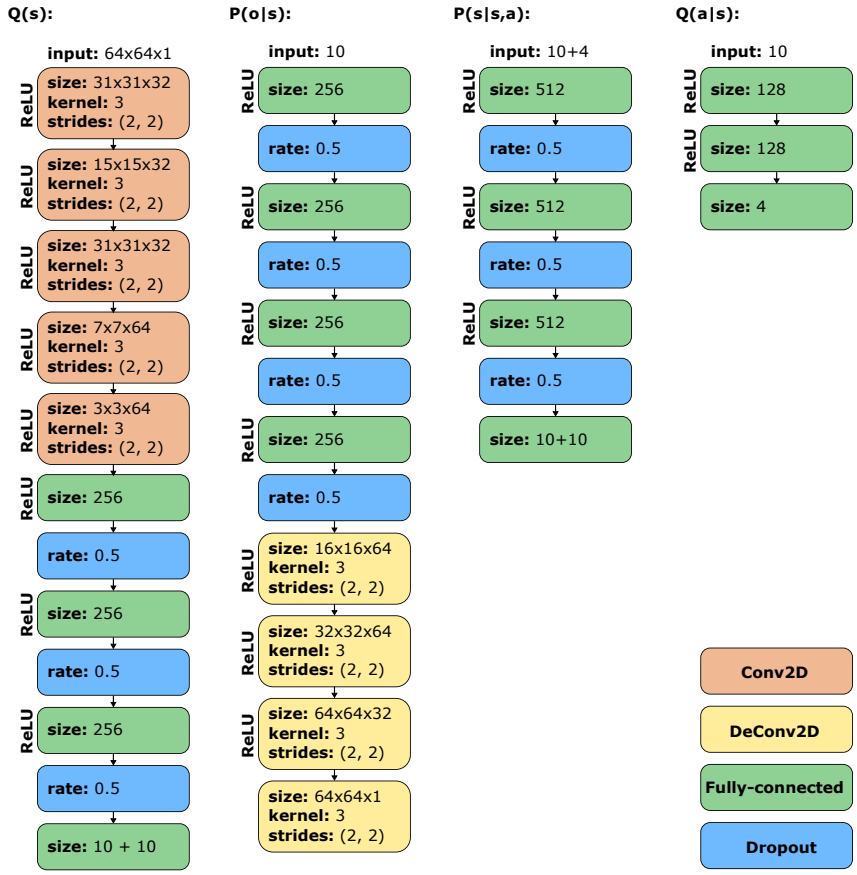

Figure 5: Neural network parameters used for the dynamic dSprites experiments. For the Animal-AI experiments, the only differences are: i) the input layer of the network used for $Q_{\phi_s}(s)$ and output layer for $P(o|s;\theta_o)$ have shape $(32, 32, 3)$, ii) the input layer of $P(s_{t+1}|s_t, a_t;\theta_s)$ has shape $(10+3)$ and iii) the output layer of $Q_{\phi_a}(a)$ has a shape of (3), corresponding to the three actions forward, left and right.

7.4 Examples of agent plans

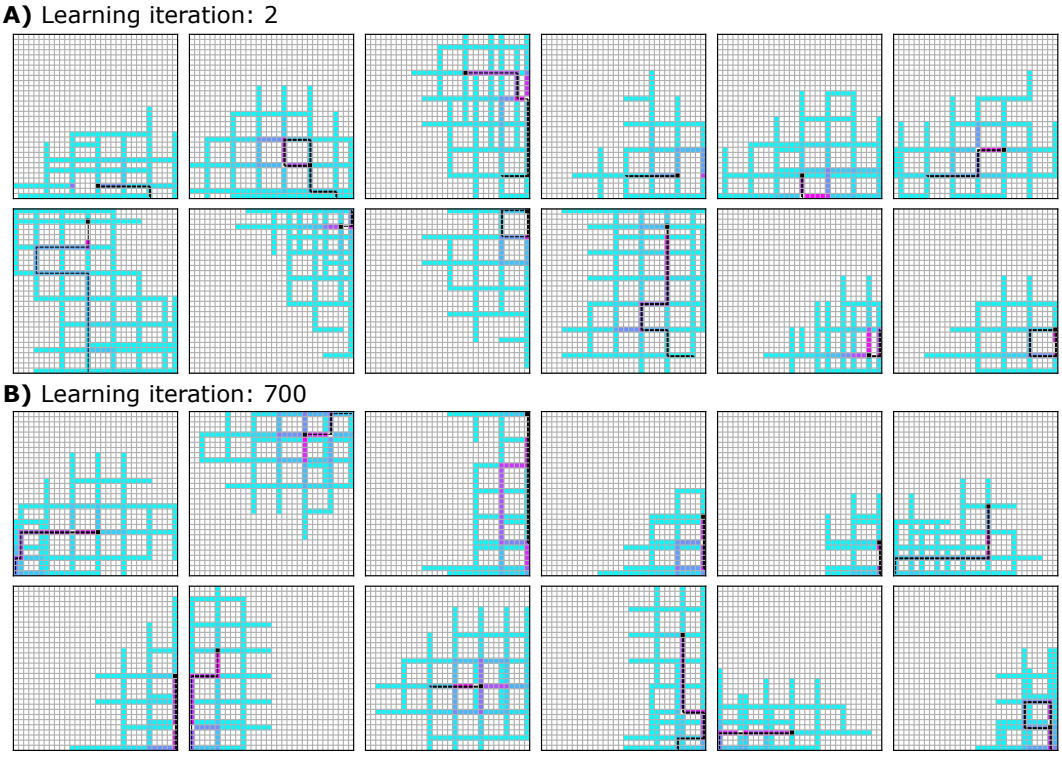

Figure 6: Examples of consecutive plans in the dynamic dSprites environment during a single experiment.

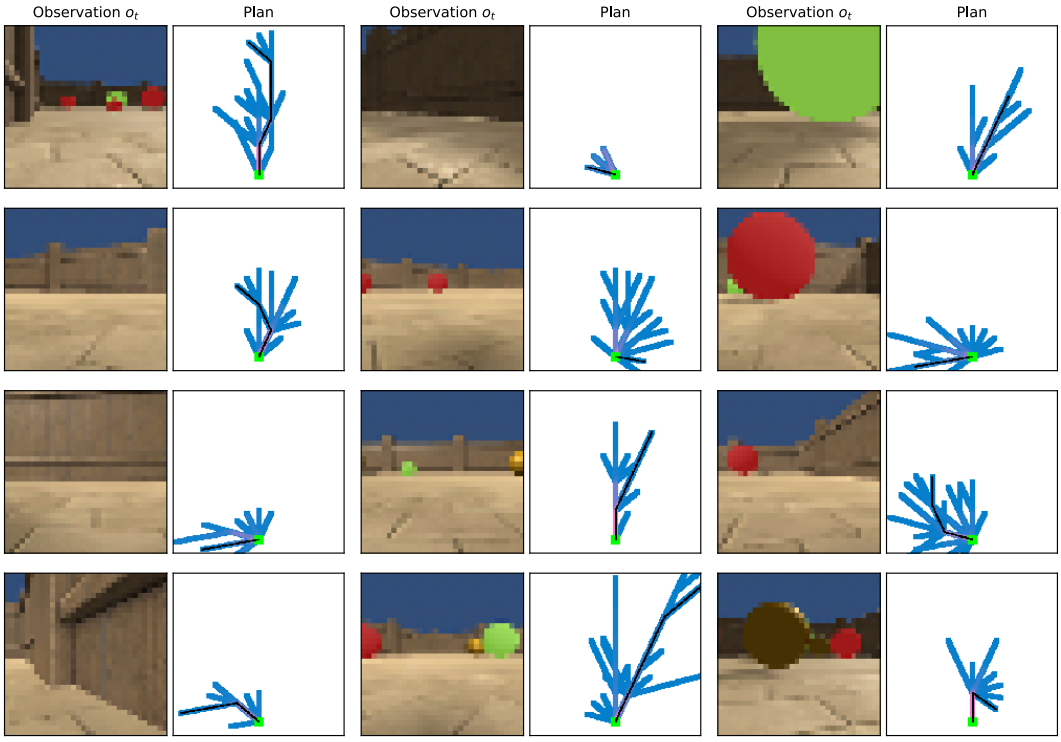

Figure 7: Examples of plans in Animal-AI environment. Examples were picked randomly.

7.5 Examples of traversals

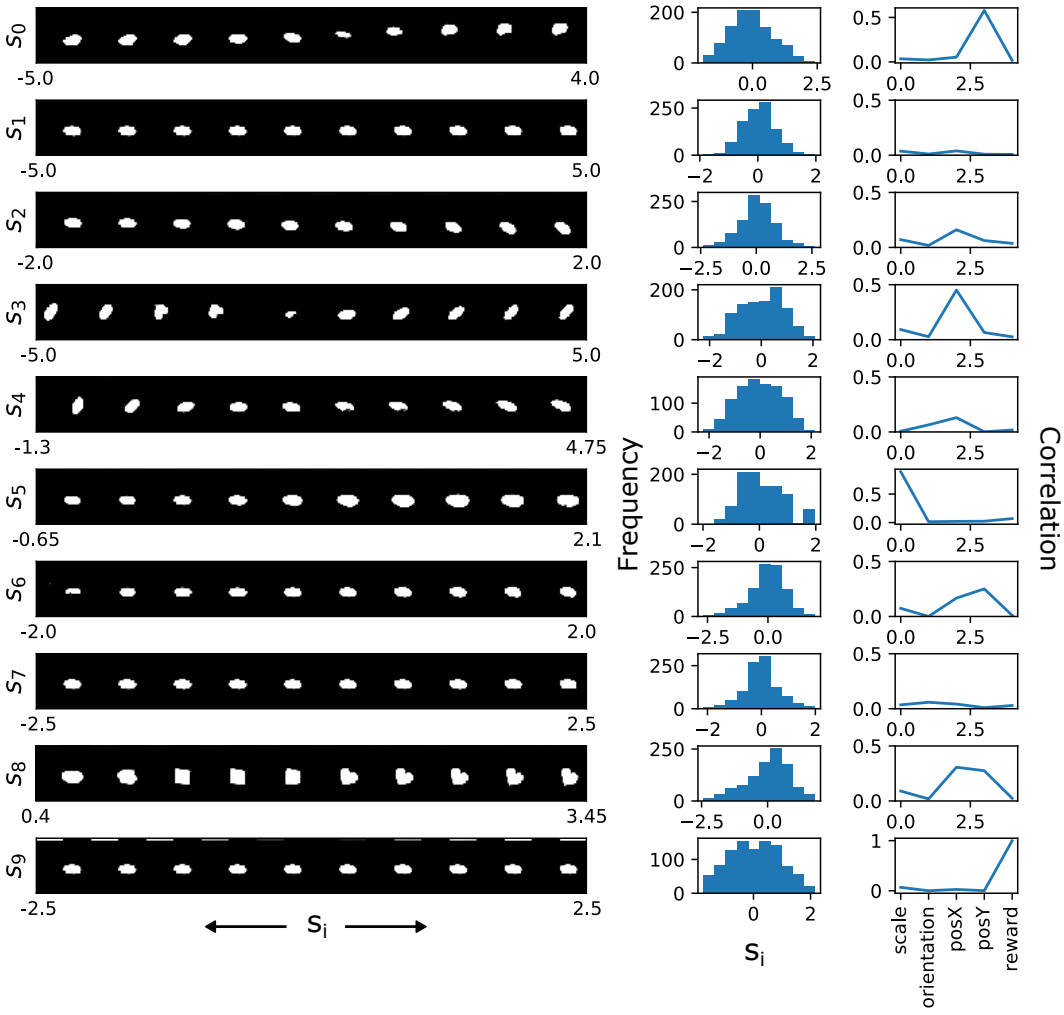

Figure 8: Latent space traversals for the full active inference agent trained in the dynamic dSprites environment. Histograms represent distribution of values for 1000 random observations. The graphs on the right column represent Spearman's correlation between each dimension of s and the 5 continuous ground truth features of the environment. This includes the 4 continuous features of the dSprites dataset and reward, encoded in the top pixels shown in s_9. Although correlation with shape types is not shown, as this is a categorical feature, the traversals of s_8 illustrate that this dimension encodes shape.