Exploration and preference satisfaction trade-off in reward-free learning

Noor Sajid [1] **Panagiotis Tigas** [2] **Alexey Zakharov** [3] **Zafeirios Fountas** [1] **Karl Friston** [1]

Abstract

Biological agents have meaningful interactions with their environment despite the absence of immediate reward signals. In such instances, the agent can learn preferred modes of behaviour that lead to predictable states – necessary for survival. In this paper, we pursue the notion that this learnt behaviour can be a consequence of reward-free preference learning that ensures an appropriate trade-off between exploration and preference satisfaction. For this, we introduce a model-based Bayesian agent equipped with a preference learning mechanism (**pepper**) using conjugate priors. These conjugate priors are used to augment the expected free energy planner for learning preferences over states (or outcomes) across time. Importantly, Pepper enables the agent to learn preferences that encourage adaptive behaviour at test time. We illustrate this in the OpenAI Gym FrozenLake and the 3D mini-world environments – with and without volatility. Given a constant environment, these agents learn confident (i.e., precise) preferences and act to satisfy them. Conversely, in a volatile setting, perpetual preference uncertainty maintains exploratory behaviour. Our experiments suggest that learnable (reward-free) preferences entail a trade-off between exploration and preference satisfaction. Pepper offers a straightforward framework suitable for designing adaptive agents, when reward functions cannot be predefined as in real environments.

[1] WCHN, University College London, UK [2]OATML, Oxford University, UK [3]Imperial College London, UK. Correspondence to: Noor Sajid <noor.sajid.18@ucl.ac.uk>.

Proceedings of the Unsupervised Reinforcement Learning Workshop at the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

1. Introduction

Extrinsic rewards are not necessary to characterise an agent's interaction with its environment. For example, humans have been shown to rely on intrinsic motivation [47, 43, 6, 72], that can adequately regulate behaviour[1]. Consequently, in the absence of immediate rewards, there is a preferred exchange with the environment [2, 53, 15], that can be updated under changing circumstances. Interestingly, this can result in accruing preferences – and habits – that may be at odds with objective goals, e.g., kleptomania. In this paper, we demonstrate that this kind of behaviour can be a consequence of reward-free preference learning that encourages self-evidencing[29] and maintains an appropriate arbitration between exploration and preference satisfaction. In brief, we will see that agents learn to explore or exploit, depending upon the predictability of environmental contingencies.

Preference satisfaction subsumes homeostatic (extrinsic) motivations that encourage individuals to maintain some 'preferred' behaviour [43] and resist effects of perturbations (external or otherwise). Generally, these refer to base needs that can be satisfied, e.g., going to sleep or eating food. Conversely, exploration involves heterostatic (intrinsic) motivations that distract the agent from its homeostatic imperatives, e.g., novelty-seeking behaviour. Exploratory behaviours would include trying a new hobby or taking a different route to work. This kind of exploration is distinct from random behaviour because it depends upon what the agent does not know. Interestingly, over time, exploration can become the primary mode of behaviour if exploration satisfies the agent's (learnt) preferences when dealing with an uncertain environment.

Our work is based on the notion that an adaptive agent learns the preferences that best reflect its environment. To this end, we present **pepper**; a preference learning mechanism that can accumulate preferences over states (or outcomes) using conjugate priors – given a model-

[1]Explicitly, this prescribes Bayes-optimal behaviour in the sense of Bayesian design and active learning.

based Bayesian agent. Here, we instantiate Pepper as a deep active inference agent [17, 69, 10, 68, 14], maximising the evidence lower bound (or minimising the free energy) during training, and optimising the expected free energy (or free energy of future trajectories) for planning [44, 40]. However, it would be straightforward to add Pepper to other Bayesian reinforcement learning (RL) agents instead e.g., MaxEnt[20], Max[57] or Dreamer[25], etc. Active inference was chosen deliberately to leverage the expected free energy (EFE) as a planning objective that captures the imperative to maximise: a) intrinsic value – from interactions with the environment – about latent states, and b) extrinsic value, namely, realising prior preferences over outcomes. Moreover, active inference's Bayesian formulation provides a natural way to introduce conjugate priors necessary for amortised learning of preferences over states (or outcomes) – as previously shown in a simplified setting for outcome preference learning[49].

Briefly, Pepper comprises a two-step procedure which occurs after the (generative) model of the agent is optimised for the environment (i.e., training time – see Fig. 1). The first step consists of short episodes of direct exchange with the environment, where a history of observations and latent state representations are retained. Once each episode finishes, the second step involves updating prior preferences based on the history using simple update rules (see Section 4). Importantly, this means that agent can learn (different) preferences that encourage adaptive behaviour at test.

The key contributions of this work are:

- We present a simple, and flexible, preference learning mechanism (Pepper) to augment the planning objective (i.e., EFE) for learning (state or outcome) preferences using deep learning.
- Pepper is reward-free at train and test time. This is achieved by casting rewards as a random variable in our generative model; equivalent to any other observation.
- Adaptive behaviour is conceptualised as a trade-off between exploration and preference satisfaction.

In what follows, we review the related literature. Next, we introduce the problem setting and pepper (the preference learning mechanism). We then evaluate the different types of preferences learnt during test time, and how they engender an appropriate trade-off between exploration and preference satisfaction. Finally, we discuss the potential implications of this work.

2. Related work

RL is regarded as a suitable framework for building artificial agents. However, by definition, it relies on a reward signal to reinforce agent behaviour [66]. In reality, agents do not operate in a problem-solving setting, where a "critic" may not be readily available to provide immediate rewards [5, 59, 60]. Without task-specific reward signal (also called *extrinsic* reward), the agent is driven by intrinsic motivations that promote exploration, play and curiosity [47, 58, 59]. Over the years, a variety of intrinsic motivation methods have been proposed, largely focusing on exploration, based on information gain [30, 64], prediction error [1, 45, 61], novelty search [39, 56], curiosity [50, 51], entropy [21, 38], or empowerment [36, 42].

Lately, through the popularisation of self-supervised learning (SSL) methods [22, 41], the deep RL community has turned its attention to self-supervised reinforcement learning. Auxiliary tasks or rewards [31] are used – in the absence of any extrinsic rewards during train time – to train intrinsically motivated agents for representation learning [70, 19, 35, 65] or generative model learning [57, 55, 3]. Recent work [32, 71] has focused on theoretical properties of reward-free Reinforcement Learning. However, the ultimate goal of such methods is to yield easily transferable representations to be exploited upon introduction of a task.

Our approach differs in several ways. First, we formulate intrinsic motivation using the Expected Free Energy [17, 33] and focus on investigating the behaviour of intrinsically motivated agents. Now exploration is simply an emergent behaviour of the planning objective and not a mechanism for improving future task performance. Second, Pepper can be used to learn preferences over both states and outcomes in a deep learning setting – extending previous formulations to high-dimensional spaces [49]. Conceptually, open-ended learning [52, 62, 63], where agents are responsible for never-ending learning opportunities, is closest to our formulation. However, Pepper extends this scenario to show how agents can trade-off between actively looking for opportunities to learn but also enjoy moments of preference satisfaction.

3. Problem setting

We consider a world that can be represented as a discrete-time Markov decision process (MDP), formally defined as a tuple of finite sets $(\mathcal{S}, \Pi, \Omega, \mathcal{P}, \mathcal{R})$, such that: $s \in \mathcal{S}$ is a particular latent state, $o \in \Omega$ is a particular image observation, $r \in \mathcal{R}$ is a particular reward observation, and $\mathcal{P}$ is a set of transition probabilities. Notice, we cast the reward function as another

random variable – no different to an image observation. Further, $\pi \in \Pi$ where $\pi = \{a_1, a_2, ..., a_T\}$ is a policy (i.e., action trajectory) and Π a finite set of all possible policies up to a given time horizon $T \in \mathbb{N}^+$ and $\mathbb{T} = \{0, .., t, .., \tau, T\}$ a finite set which stands for discrete time; t the current time and τ some future time. In short, we do not assume an optimal state-action policy but consider sequential policy optimisation inherent in active inference. Accordingly, the agent's generative model is defined as a probability density, $P_\theta(o, r, s, \pi)$, parameterised by θ (Fig. 1).

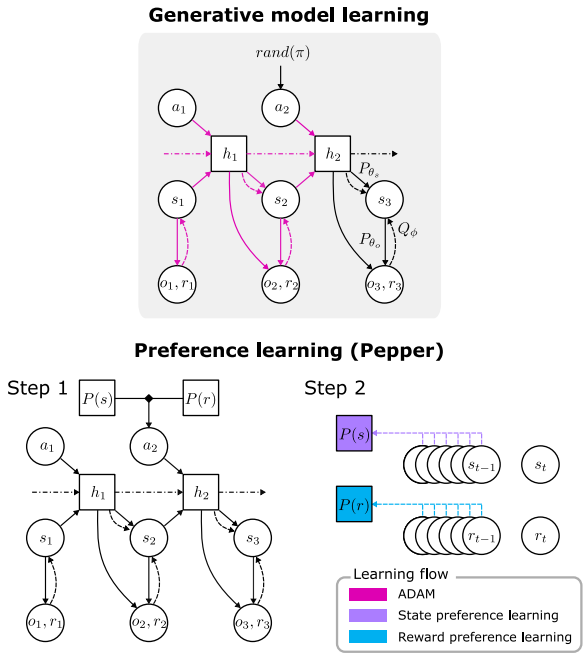

Figure 1. Model architecture and 2-step training procedure. Circles and squares denote random and deterministic variables respectively. Coloured lines denote connections where learning is employed. Shaded circles represent outcomes that have already been observed by the agent. The first figure shows the generative model used during learning. The second panel is for Pepper (the preference learning phase) – comprising two steps in each episode: 1) interaction with the environment, & 2) accumulation of preferences once interaction ends. There is a bi-directional flow between the 2 steps: step 1 influences preference learning and step 2 in turn influences environment interaction in the next episode.

The generative model is instantiated as a Recurrent State-Space Model (RSSM) [24, 23, 25][2] where a history of observations $(o_0, o_1, .., o_t)$ and actions $(a_0, a_1, .., a_t)$ are mapped to a sequence of deterministic states h_t. Using these, distributions over the latent states – both prior and posterior – can be attained. Formally, this consists of the following: 1) a GRU [12] based deterministic recurrent model, $h_t = f_\theta(h_{<t}, s_{<t}, \pi_{<t})$; 2) latent state posterior, $Q_\phi(s_t|h_t, o_t) \sim Cat$, and prior, $P(s) \sim Cat(D)$; 3) transition model, $P_\theta(s_t|h_t) \sim Cat$; 3) image predictor (or emission model), $Q_\phi(o_t|h_t, s_t) \sim Bernoulli$; and 4) reward model, $Q_\phi(r_t|h_t, s_t)$, and prior, $P(r) \sim Cat(C)$; where $Q_\phi(\cdot)$ denotes the approximate distribution, parameterised by ϕ.

[2] https://github.com/danijar/dreamerv2 (MIT License)

3.1. Learning the generative model

To learn the generative model the evidence lower bound ($ELBO$) of the likelihood $p(o_{1:T}, r_{1:T} \mid \pi)$ [24] or equivalently, the variational free energy [16, 14, 49] was optimised:

$$\mathcal{L}(\theta) = \sum_{t=1}^{T} \Big[\underbrace{-\mathbb{E}_{Q_\phi}\left[\ln P_\theta(o_t \mid s_t, \pi) + \ln P_\theta(r_t \mid s_t, \pi)\right]}_{\text{reconstruction}} \tag{1}$$

$$+ \underbrace{\mathbb{E}_{Q_\phi}\left[D_{KL}(Q_\phi \parallel P_\theta(s_t \mid s_{t-1}, \pi)\right]}_{\text{dynamics}} \Big] \tag{2}$$

where, D_{KL} denotes the Kullback–Leibler divergence. Practically, this entails using trajectories generated under a random policy. See Appendix A for $ELBO$ implementation details.

4. Pepper: preference learning mechanism

After learning the generative model, we substitute the planning objective with the expected free energy. At time-step t and for a time horizon up to time T, the expected free energy (EFE) is [17, 14]:

$$G(\pi) = \sum_{\tau=t}^{T} \mathbb{E}_{\tilde{Q}}\left[\log Q_\phi(s_\tau|\pi) - \log \tilde{P}_\theta(o_\tau, s_\tau|\pi)\right] \tag{3}$$

where $\tilde{Q} = Q_\phi(o_\tau, s_\tau, \theta|\pi) = Q(\theta|\pi)Q(s_\tau|, \pi)Q_\phi(o_\tau|s_\tau, \pi)$ and $\tilde{P}_\theta(o_\tau, s_\tau|\pi) = P(o_\tau)Q(s_\tau|o_\tau)P(\theta|s_\tau, o_\tau, \pi)$. This is an appropriate planning objective because it: 1) is analogous to the expectation of the ELBO (Eq.1) under the predictive posterior $P_\theta(o_t|s_t, \pi)$ and 2) can be decomposed into extrinsic and intrinsic value without any additional terms [13, 40]. Accordingly, in the absence of learnt preferences – or whilst learning them – intrinsic motivation contextualises agent's interactions with the environment environment in a way that depends upon its posterior beliefs about latent environmental states [4, 47]. Actions are selected by sampling from

distribution $P(\pi) = \arg\max(-G(\pi))$. We Pepper this planning objective with conjugate priors to allow for preference learning over time.

4.1. Learning preferences using conjugate priors

A natural way to learn preferences is to extend the agent's generative model with conjugate priors over prior beliefs (i.e., hyper-priors) for each appropriate probability distributions, that are learnt over time [17, 49, 13]. Generally, for closed-form updates any exponential family would be appropriate [46]. Since our distributions of interest, latent state and rewards are Categorical, we used the Dirichlet distribution as the conjugate prior. For the latent state, $P(s) \sim Cat(D)$, this is defined as:

$$P(D^i|d^i) = Dir(d^i) \Rightarrow \mathbb{E}_{P(D^i|d^i)}\left[D^i_{ij}\right] = \frac{d^i_{ij}}{\sum_k d^i_{kj}} \quad (4)$$

where, $\digamma$ is the digamma function, $d \sim \mathbb{R}^+$ and same parameterisation holds for $P(r) \sim Cat(C)$. The posteriors for the Dirichlet hyper–parameters are evaluated by updating the prior using the following rule $\boldsymbol{d_{i,j}} + \alpha * s_{i,j}$ where $s_{i,j}$ are the observations for that particular category and α the learning rate. These estimates can be treated as pseudo–counts, and the ensuing learning procedure is reminiscent of Hebbian plasticity – see [17, 48] for discussion. In effect, this allows the agent to accumulate contingencies and learn about what it prefers – either via outcomes or states.

4.1.1. Augmented expected free energy

To incorporate preference over both states and reward outcomes, we use two distinct EFE decompositions. The first one instantiates preference learning over reward outcomes, as presented below for a single time instance τ [54, 14]:

$$G(\pi,\tau) = -\mathbb{E}_{\tilde{Q}}\left[\log P(r|C)\right] \quad (5a)$$
$$+\mathbb{E}_{\tilde{Q}}\left[\log Q(s_\tau|\pi) - \log P(s_\tau|o_\tau,\pi)\right] \quad (5b)$$
$$+\mathbb{E}_{\tilde{Q}}\left[\log Q(\theta|s_\tau,\pi) - \log P(\theta|s_\tau,o_\tau,\pi)\right] \quad (5c)$$

where, $P(r|C)$ is the probability of a particular reward outcome given (learnt) prior preferences (C). The second decomposition incorporates preferences over latent states:

$$G(\pi,\tau) = -\mathbb{E}_{\tilde{Q}}\left[\log P(o_\tau|s_\tau,\pi)\right] \quad (6a)$$
$$+\mathbb{E}_{\tilde{Q}}\left[\log Q(s_\tau|\pi) - \log P(s|D)\right] \quad (6b)$$
$$+\mathbb{E}_{\tilde{Q}}\left[\log Q(\theta|s_\tau,\pi) - \log P(\theta|s_\tau,o_\tau,\pi)\right] \quad (6c)$$

where, $P(s|D)$ is the probability of a particular state given (learnt) prior preferences (D). These prior distributions are read as 'preferences' in active inference [49] – in the sense they are the outcomes the agent expects its plans to secure. For both formulations, we drop the conditioning on policy when learning of preferences and the requisite probability is calculated using Thompson sampling. This entails sampling from the prior Dirichlet distribution and estimating the likelihood. Additionally, two of the three terms that constitute the expected free energy cannot be easily computed as written in Eq.5 & Eq.6. To finesse their computation, we rearrange these expressions and use deep ensembles [37] to render these expressions tractable. See Appendix A for implementation details.

Algorithm 1 Pepper

Input :
$h_t := f_\theta(h_{<t}, s_{<t}, \pi_{<t})$ Recurrent model
$Q_\phi(s_t|h_t, o_t)$ Posterior model
$Q_\phi(s_t|h_t)$ Prior model
$P_\theta(o_t|h_t, s_t)$ Observation model
$P_\theta(r_t|h_t, s_t)$ Reward model
Initialise uniform Dirichlet prior over $P(r)$ or $P(s)$
`/* prior preference being learnt */`
learning rate α **for** each episode e **do**
  reset environment and collect initial observations (o_0 or r_0) **for** each time step t **do**
    compute $s_{po} \sim Q_\phi(s_t|h_t, o_t)$ or $s_{pr} \sim Q_\phi(s_t|h_t)$ compute G (Eq.5 or Eq.6) using (learnt) priors, observed and predicted posteriors $a_t \leftarrow \arg\max(-G(\pi))$ **execute** a_t and receive o or r $o_{t+1} \leftarrow o$ and $r_{t+1} \leftarrow r$
  end
  if reward preference learning **then**
    $c_{i,t} \leftarrow c_{i,t-1} + \alpha * r_{i,t}$ `/* Update rule for` $dir(c)$ `*/`
  else if state preference learning **then**
    $d_{ij,t} \leftarrow d_{ij,t-1} + \alpha * s_{pr_{ij,t}}$ `/* Update rule for` $dir(d)$ `*/`
end

4.1.2. Pepper

Pepper comprises a double loop during preference learning. The first loop is across time-steps when the agent interacts with the environment. This loop stores information about what happened (including observations, rewards, posterior, prior, etc). The second loop, evolving at a slower timescale, is across episodes and entails preference learning. Specifically, once the interaction with the environment ends, the agent updates the prior Dirichlet distribution (over preferences) using the data gathered during the episode. In the subsequent time-

steps the updated preferences are used to select the next action (via their influence on expected free energy). The data gathered during this episode are used to update preferences. In turn, these preferences are used to select actions during the next episode, and so on. This demonstrates a bi-directional flow between the two loops: information from environment interactions influences preference learning and the learnt preferences influence environment interaction in the subsequent episode. The Pepper preference learning procedure is summarised in Algorithm.4.1.1.

5. Results

Here, we present two sets of numerical experiments that underwrite the face validity of pepper in two and three-dimensional environments, respectively.

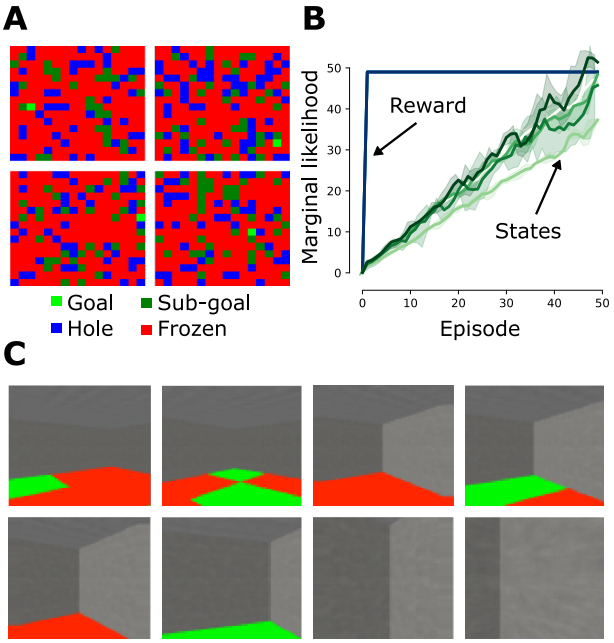

Figure 2. **A:** The graphics show examples of the different OpenAI Gym FrozenLake $16x16$ environments used. **B:** The line plots shows the marginal likelihood (y-axis), across 50 preference learning episodes (x-axis), for $P(s)$ (i.e., states) and $P(r)$ (i.e., reward) during test time. Here, the dark lines represent the mean (across 10 seeds), and shaded area the 95% confidence interval. Different shades of green denote levels of environment volatility. **C:** The graphics show a particular agent trajectory across the $3D-$ TileWorld environment – with a $10-$step interval between each image.

5.1. FrozenLake

We used a variation of the OpenAI Gym [8][3] FrozenLake environment to: 1) evaluate different behaviours acquired (at test time) when either $P(s)$ or $P(r)$ was learnt, 2) qualify how preferences can evolve as a result of environmental volatility, and 3) quantify the trade-off between exploration and preference satisfaction. The agent in the original FrozenLake formulation is tasked with navigating a grid world comprised of frozen, hole and goal tiles, using 4 actions (left, right, down or up). The agent receives a reward of 10 upon reaching the goal and a penalty -0.25 upon moving to the hole. To test preference learning, we included a sub-goal tile and removed the reward signal (Fig.2A). In other words, although the preference learning agent can differentiate between tile categories – given its generative model – it receives no extrinsic signal from the environment. Here, we simulated a volatile environment by switching the FrozenLake tile configuration every K steps and initialising the agent in a different location at the start of each episode. This furnished an appropriate test-bed to assess how much volatility was necessary to induce exploratory behaviour and shifts in learnt prior preference.

5.2. TileWorld

We extended the FrozenLake environment in the miniworld framework [11][4] to three dimensions to test the generalisation and scalability of preference learning, when operating in a 3D visual world (Fig.2C). In this task, the agent moves around in a small room with grey walls, frozen and goal tiles on the floor. The agent spawns in a random location and receives pixel observations (32x32 pixels with RGB channels) and a scalar value (reward) containing some information regarding the tile its currently on (1 for red tiles, 2 for green tiles). Additionally, we introduce environmental volatility by changing the floor tiles to a random map and back to the original map every K steps. Alternating between the original and a random map every K step is important to promote exploratory, novelty-seeking behaviour.

In the experiments that follow, we test agent behaviour in the two environments, with and without volatility. The Dirichlet distribution for either prior preference distribution, $P(s)$ or $P(r)$, was initialised as 1 (i.e., uniform preferences). Trained network weights, optimised Eq.1 using ADAM [34], were frozen during these experiments. Therefore, behavioural differences are a

[3] https://github.com/openai/gym/ (MIT license)
[4] https://github.com/maximecb/gym-miniworld/ (Apache 2.0 License)

direct consequence of pepper that induces differences in estimation of the EFE. See Appendix B for architecture and training details for each environment.

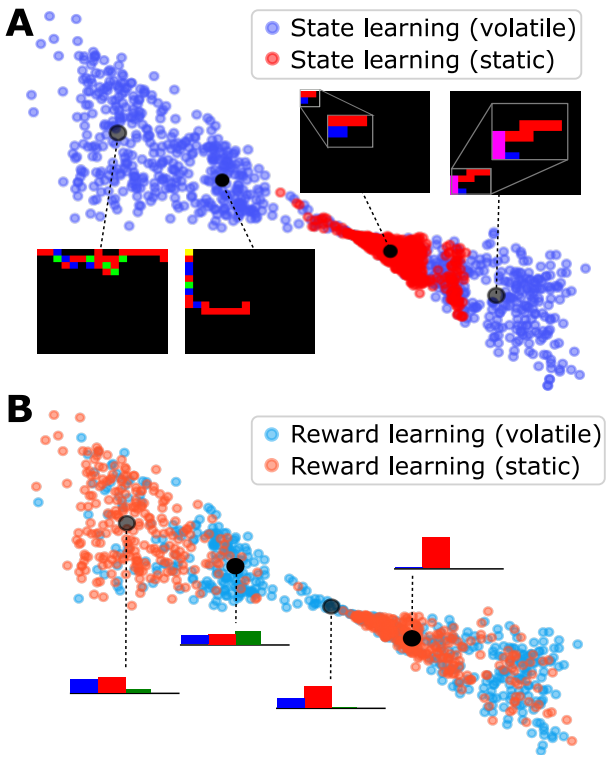

Figure 3. Visualisation of the posterior latent states (estimated using $Q_\phi(s_t|h_t, o_t)$) during preference learning of ($P(s)$ **A** and $P(r)$ **B**) episodes. The states have been projected onto the first two principle components, and the black circles represent their k-mean centroid. Here, the accompanying graphics present a representative agent trajectory (**A** with visited tiles highlighted) and reward profile (**B**) from that particular cluster.

5.3. Learnt preferences

State preferences Unsurprisingly, preference learning over latent states in the FrozenLake environment revealed two types of behaviours: exploration and preference satisfaction (Fig.3A & 4A)). Here, preference satisfaction entailed restricted movement within a small section of the grid with gradual accumulation of prior preferences (see examples in the Appendix C.1). This speaks to the self-evidencing nature of Pepper. That is as the Pepper agent sees similar observations across episodes it grows increasingly confident (via increased precision over the prior preference) that these are the outcomes it prefers. Conversely, exploratory behaviour is evident in a volatile setting (Fig.3A & 4A)), as gradual preference accumulation entails encoding of previously unseen states (see examples in Appendix C.1).

Reward preferences Reward preference learning revealed subtle differences in preferences (Fig.3B), where certain agents preferred sub-goal tiles more than neutral tiles. All pepper agents were able to immediately maximise their marginal likelihood[5] over the reward (Fig.3B). However, we did not observe clear differences in preference learning as the environment context shifted from non-volatile (i.e., map change) to increasingly volatile (map changed every time step). We consider this to be consequence of the sparse categories over the reward distribution, and the large percentage of map being taken up by the neutral tiles. This meant preference accumulation was biased in favour of the neutral tile (see examples in Appendix C.1).

5.4. Exploration and preference satisfaction trade-off

We evaluated the exploration and preference satisfaction trade-off using Hausdorff distance (Fig.4) [7]. This is an appropriate metric, which calculates the maximum distance of the agents position in a particular trajectory to the nearest position taken in another trajectory. Accordingly, a high Hausdorff distance denotes increased exploration, since trajectories observed across episodes differ from one other. Whereas, a low distance entails prior preference satisfaction as agents repeat trajectories across episodes. Using this metric, an inverted u-shaped association (Fig.4A), between volatility in the environment and preference satisfaction, is observed for preference learning over the states. Here, 50% volatility in the environment, shifts the pepper agents behaviour from satisfying preferences to becoming exploratory, when faced with an uncertain environment (and inability to predict the future). Agents in this setting (with the highest Hausdorff distance) tend to pursue long paths from the initial location. (Fig.3A).

Interestingly, at 100% environmental volatility, the Pepper agents behaviour shifts back to satisfying its preferences. These agents learn bi-modal preferences over the latent states. Irregardless of how the map changes they move directly to either location given the initial position (see trajectories in Appendix C.2). This ability to disregard random, noisy information with continuous map changes (analogous to the noisy-TV setting introduced in [9]) highlights a motivation beyond random exploration under state preference learning. For reward learning, exploration is also instantiated with increased

[5]Marginal likelihood is simply the likelihood function of the parameter of interest (here state or reward) where some parameter variables have been marginalised out.

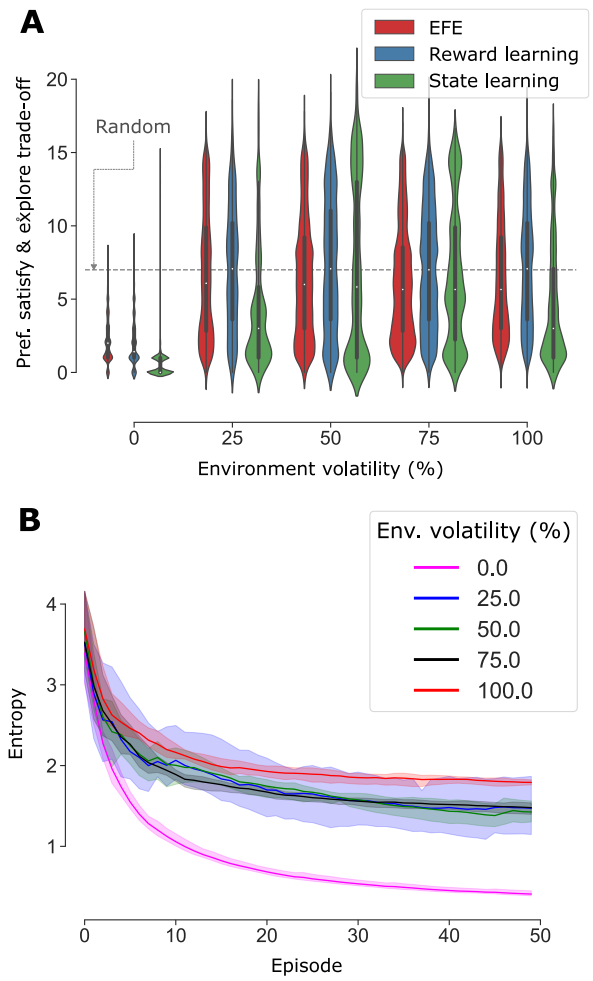

Figure 4. **A:** The violin plot presents the preference satisfaction and exploration trade-off measured using Hausdorff distance [7] at different levels of volatility in the environment. The x-axis denotes environment volatility at constant map (0%), change in map every 40 steps (25%), 20 steps (50%), 10 steps (75%) and every step (100%). The y-axis denotes the Hausdorff distance. Here, red is for the agent optimising the standard expected free energy (EFE) Eq.3, blue for reward preference learning Eq.5 and green for state preference learning Eq.6. **B:** The line plot depicts the entropy over $P(s)$ across varying levels of volatility in the environment. The x-axis represents the episodes, and the y-axis entropy (in natural units). Here, the dark lines represent the mean (across 10 seeds), and shaded area the 95% confidence interval. The pink line is for 0%, blue for 25%, green for 50%, black for 75% and red for 100% volatility in the environment.

volatility in the environment. Yet, complete volatility does not trigger a definitive shift back to preference satisfaction. Additionally, the expected free energy agent, without preference learning capacity, also exhibits a shift in behaviour as the environment becomes volatile. The exploratory behaviour here is driven exclusively by an imperative to resolve state uncertainty, (Eq.5b): i.e., the mutual information between the agent's beliefs about its latent state representation of the world, before and after making a new observation.

5.5. Preference learning in the volatile TileWorld

Preference learning agents evinced a strong preference for looking at grey walls in the volatile TileWorld environment (Fig.5A). This was consistently observed for both preference learning over latent states and rewards. Importantly, when spawned in a location right next to the wall, these agents were happy to satisfy their preferences and not move (Fig.5B). This is driven by three factors: fast preference accumulation over grey walls, a small number of state configurations and a generative model that is able to appropriately predict future trajectories (see example reconstructions and imagined roll-outs in the Appendix B.2). However, volatility in the environment does influence the encoding of prior preferences – as evident from the observed state entropy fluctuations across the episodes (Fig.5).

6. Concluding remarks

Summary: Pepper – the reward-free preference learning mechanism presented – provides a simple way to influence agent behaviour. Although, unlike the RL formulation, we cast *what is preferred* to the agent instead of the environment 'designer'. That is, the agent is responsible for interacting with the environment and over time developing preferences that it acts to satisfy without an extrinsic signal. Our experiments revealed that rich category spaces are necessary for learning preferences that can establish distinct behavioural strategies – specifically, within a volatile setting. Thus, future experiments looking to leverage *pepper* should provide a suitable category space to learn over.

Conjugate priors and Hebbian plasticity: We employed a simple learning strategy for accruing preferences, using conjugate priors. This type of learning usually calls on associative or Hebbian plasticity [26], where synaptic efficacy is reinforced by the simultaneous firing of pre-and post-synaptic neurons. For example, as more neutral tiles are observed, more evidence is accumulated in the synaptic connection to support the hypothesis that neutral tiles are preferentially observed. Implicitly, pepper rests upon experience-dependent plasticity, i.e., strengthening of synaptic

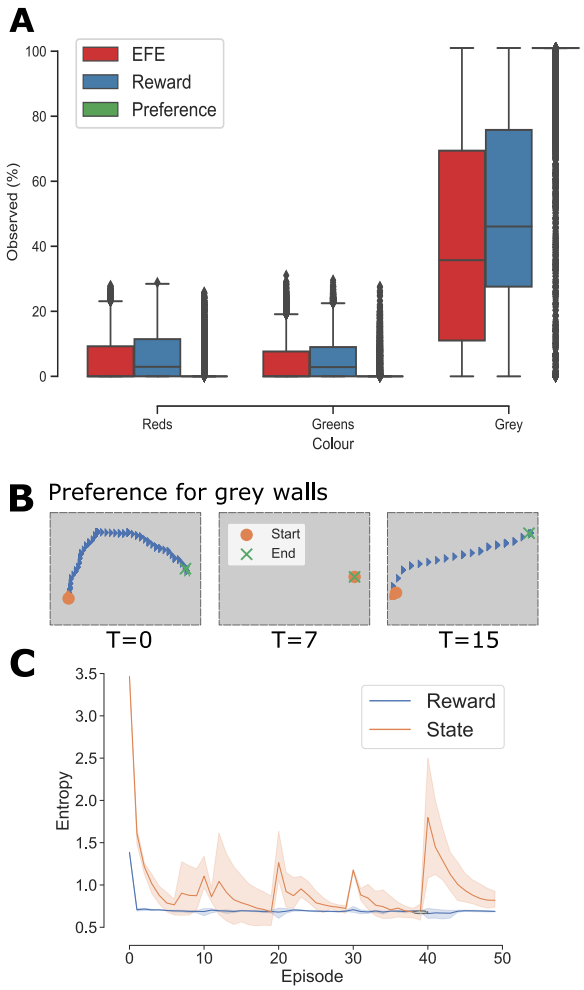

Figure 5. **A:** The bar chart plots the percentage of time the agents spent observing the 3 colours in the TileWorld. The x-axis presents the colours: red (floor tile), green (floor tile) and grey (wall colour), and y-axis the percentage of observations calculated using the 32x32x3 pixel image the agent received at each time step. Red is for the agent optimising the standard expected free energy (EFE) Eq.3, blue for reward preference learning Eq.5 and green for state preference learning Eq.6. **B:** Sample trajectories for a single agent are presented for agents acquiring a preference for observing grey walls during state preference learning. Here, an orange circle denotes starting position, a blue triangle represents the agents location until the final position, and a green cross is the final position. **C:** The line plot depicts the entropy over $P(s)$ (in orange) and $P(r)$ (in blue) across varying levels of volatility in the environment. The x-axis represents the episodes, and the y-axis entropy (in natural units). Here, the dark lines represent the mean (across 10 seeds), and shaded area the 95% confidence interval.

connections during inference. These updates can have different times scales, and experiential levels (please see Appendix D).

Additionally, our learning process is driven by synaptic plasticity that allows certain random variables (s, r) to expand in light of new experiences [18]. They are only updated after an exchange with the environment. This separation of 'experiencing' the world, and then 'updating' model parameters is reminiscent of sleep; in which synaptic homeostasis resets model parameterisation, by encoding new synapses or removing redundant ones [27, 67, 28]. Removal of redundant model parameters is evaluated in experiments presented in Appendix D, where removal of accumulated Dirichlet parameters reveals consistently exploratory agents.

Limitations: Notably, having a reward-free formulation is both an advantage and limitation of our approach. By not specifying an extrinsic reward function we forego control over the agent's behaviour. In other words, by removing the ability to manipulate agent preferences regarding what is considered rewarding leads to the removal of the only clear communication channel that can be used by a designer to control agent behaviour and/or define task goals. Therefore, it would not be appropriate to use pepper in a setting where control of the agent's behaviour is required e.g., for autonomous vehicles. And, if it were, precise preferences would have to be established under supervision.

References

[1] Joshua Achiam and S. Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *ArXiv*, abs/1703.01732, 2017.

[2] W Ross Ashby. *An introduction to cybernetics*. Chapman & Hall Ltd, 1961.

[3] Philip Ball, Jack Parker-Holder, Aldo Pacchiano, Krzysztof Choromanski, and Stephen Roberts. Ready policy one: World building through active learning. In *International Conference on Machine Learning*, pages 591–601. PMLR, 2020.

[4] Andrew G Barto. Intrinsic motivation and reinforcement learning. In *Intrinsically motivated learning in natural and artificial systems*, pages 17–47. Springer, 2013.

[5] Andrew G Barto, Satinder Singh, and Nuttapong Chentanez. Intrinsically motivated learning of hierarchical collections of skills. In *Proceedings of the 3rd International Conference on Development and Learning*, pages 112–19. Piscataway, NJ, 2004.

[6] Daniel E Berlyne. Conflict, arousal, and curiosity. 1960.

[7] Henry Blumberg. Hausdorff's grundzüge der mengenlehre. *Bulletin of the American Mathematical Society*, 27(3):116–129, 1920.

[8] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[9] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

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

[11] Maxime Chevalier-Boisvert. gym-miniworld environment for openai gym. https://github.com/maximecb/gym-miniworld, 2018.

[12] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[13] Lancelot Da Costa, Thomas Parr, Noor Sajid, Sebastijan Veselic, Victorita Neacsu, and Karl Friston. Active inference on discrete state-spaces: a synthesis. *Journal of Mathematical Psychology*, 99:102447, 2020.

[14] Zafeirios Fountas, Noor Sajid, Pedro AM Mediano, and Karl Friston. Deep active inference agents using monte-carlo methods. *arXiv preprint arXiv:2006.04176*, 2020.

[15] Karl Friston. Life as we know it. *Journal of the Royal Society Interface*, 10(86):20130475, 2013.

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

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

[18] Min Fu and Yi Zuo. Experience-dependent structural plasticity in the cortex. *Trends in neurosciences*, 34(4):177–187, 2011.

[19] Zhaohan Daniel Guo, M. G. Azar, Bilal Piot, B. A. Pires, Tobias Pohlen, and R. Munos. Neural predictive belief representations. *ArXiv*, abs/1811.06407, 2018.

[20] Tuomas Haarnoja, Sehoon Ha, Aurick Zhou, Jie Tan, George Tucker, and Sergey Levine. Learning to walk via deep reinforcement learning. *arXiv preprint arXiv:1812.11103*, 2018.

[21] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

[22] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[23] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[24] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

[25] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[26] Donald Olding Hebb. The organization of behavior; a neuropsycholocigal theory. *A Wiley Book in Clinical Psychology*, 62:78, 1949.

[27] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268 (5214):1158–1161, 1995.

[28] J Allan Hobson and Karl J Friston. Waking and dreaming consciousness: neurobiological and functional considerations. *Progress in neurobiology*, 98 (1):82–98, 2012.

[29] Jakob Hohwy. The self-evidencing brain. *Noûs*, 50(2):259–285, 2016.

[30] Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *NIPS*, 2016.

[31] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[32] Chi Jin, Akshay Krishnamurthy, Max Simchowitz, and Tiancheng Yu. Reward-free exploration for reinforcement learning. In *International Conference on Machine Learning*, pages 4870–4879. PMLR, 2020.

[33] Raphael Kaplan and Karl J Friston. Planning and navigation as active inference. *Biological cybernetics*, 112(4):323–343, 2018.

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

[35] Thomas Kipf, Elise van der Pol, and M. Welling. Contrastive learning of structured world models. *ArXiv*, abs/1911.12247, 2020.

[36] Alexander S Klyubin, Daniel Polani, and Chrystopher L Nehaniv. Empowerment: A universal agent-centric measure of control. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 128–135. IEEE, 2005.

[37] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

[38] Lisa Lee, Benjamin Eysenbach, Emilio Parisotto, Eric P. Xing, Sergey Levine, and Ruslan Salakhutdinov. Efficient exploration via state marginal matching. *ArXiv*, abs/1906.05274, 2019.

[39] J. Lehman and K. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In *ALIFE*, 2008.

[40] Beren Millidge, Alexander Tschantz, and Christopher L. Buckley. Whence the Expected Free Energy? *Neural Computation*, 33(2):447–482, 02 2021. ISSN 0899-7667. doi: 10.1162/neco_a_01354. URL https://doi.org/10.1162/neco_a_01354.

[41] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6707–6717, 2020.

[42] Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *NIPS*, 2015.

[43] Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in Neurorobotics*, 1:6, 2009. ISSN 1662-5218. doi: 10.3389/neuro.12.006.2007. URL https://www.frontiersin.org/article/10.3389/neuro.12.006.2007.

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

[45] Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 488–489, 2017.

[46] H. Raiffa and R. Schlaifer. *Applied Statistical Decision Theory*. Studies in managerial economics. Division of Research, Graduate School of Business Administration, Harvard University, 1961. URL https://books.google.co.uk/books?id=SpOOKFcFQDsC.

[47] Richard M Ryan and Edward L Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, 25(1):54–67, 2000.

[48] Noor Sajid, Thomas Parr, Thomas M Hope, Cathy J Price, and Karl J Friston. Degeneracy and redundancy in active inference. *Cerebral Cortex*, 30(11):5750–5766, 2020.

[49] Noor Sajid, Philip J Ball, Thomas Parr, and Karl J Friston. Active inference: demystified and compared. *Neural Computation*, 33(3):674–712, 2021.

[50] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. 1991.

[51] Jürgen Schmidhuber. Simple algorithmic principles of discovery, subjective beauty, selective attention, curiosity & creativity. In *Discovery Science*, 2007.

[52] Jürgen Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

[53] Roger Schrodinger, Erwin Schrödinger, and Erwin Schr Dinger. *What is life?: With mind and matter and autobiographical sketches*. Cambridge University Press, 1992.

[54] 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.

[55] Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pages 8583–8592. PMLR, 2020.

[56] Pranav Shyam, Wojciech Jaskowski, and Faustino Gomez. Model-based active exploration. In *ICML*, 2019.

[57] Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. In *International Conference on Machine Learning*, pages 5779–5788. PMLR, 2019.

[58] Satinder Singh, A. Barto, and N. Chentanez. Intrinsically motivated reinforcement learning. In *NIPS*, 2004.

[59] Satinder Singh, Andrew G Barto, and Nuttapong Chentanez. Intrinsically motivated reinforcement learning. Technical report, MASSACHUSETTS UNIV AMHERST DEPT OF COMPUTER SCIENCE, 2005.

[60] Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

[61] Bradly C. Stadie, Sergey Levine, and P. Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *NIPS*, 2015.

[62] Russell K Standish. Open-ended artificial evolution. *International Journal of Computational Intelligence and Applications*, 3(02):167–175, 2003.

[63] Kenneth O Stanley, Joel Lehman, and Lisa Soros. Open-endedness: The last grand challenge you've never heard of. *While open-endedness could be a force for discovering intelligence, it could also be a component of AI itself*, 2017.

[64] Susanne Still and Doina Precup. An information-theoretic approach to curiosity-driven reinforcement learning. *Theory in Biosciences*, 131:139–148, 2011.

[65] Adam Stooke, Kimin Lee, P. Abbeel, and M. Laskin. Decoupling representation learning from reinforcement learning. *ArXiv*, abs/2009.08319, 2020.

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

[67] Giulio Tononi and Chiara Cirelli. Sleep function and synaptic homeostasis. *Sleep medicine reviews*, 10(1):49–62, 2006.

[68] Alexander Tschantz, Anil K Seth, and Christopher L Buckley. Learning action-oriented models through active inference. *PLoS computational biology*, 16(4):e1007805, 2020.

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

[70] Aäron van den Oord, Y. Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

[71] Ruosong Wang, Simon S Du, Lin F Yang, and Ruslan Salakhutdinov. On reward-free reinforcement learning with linear function approximation. *arXiv preprint arXiv:2006.11274*, 2020.

[72] Robert C Wilson, Andra Geana, John M White, Elliot A Ludvig, and Jonathan D Cohen. Humans use directed and random exploration to solve the explore–exploit dilemma. *Journal of Experimental Psychology: General*, 143(6):2074, 2014.

A. Pepper implementation

Pepper was implemented as an extension to Dreamer V2 [25] public implementation[6]. Specifically, Dreamer's generative model training loop was used, alongside a model predictive control (MPC) planner. Therefore, the actor learning part of Dreamer was not incorporated, and the generative model was trained using Plan2Explore [55]. Like Plan2Explore, an ensemble of image encoders were learnt and the "disagreement" of the encoders was used as an intrinsic reward during training. This guides the agent to explore areas of the map that have high novelty and potentially high information gain, when acquiring a generative (i.e., world) model. Unfortunately, replacing the amortised policy with this planner made the environment interaction (i.e., the acting loop) relatively slow. We implemented the planner as described in Algorithm A.

Algorithm 2 Planner

Input :
s_t current state
N Number of random action sequences to evaluate
Initialise **for** i=1...N **do**
  $\pi^i \sim \mathcal{U}$ `/* Random action */`
  $\text{score}_i \leftarrow 0$ **for** $\tau = t \dots H$ **do**
    $\text{score}_i \leftarrow \text{score}_i - G(\pi^i, \tau)$ `/* Updated according Eq.5 or Eq.6 */`
  end
end
$k \leftarrow \arg\max \text{score}$ **Return** π^k

Upon training completion, we froze the generative model learnt weights and only allowed learning of prior preferences. These (state or reward) preferences were updated after each episode, as described in Algorithm 4.1.1.

A.1. Evidence lower bound

The generative model was optimised using the ELBO formulation introduced in [23]:

$$\mathcal{L}(\theta) = \sum_{t=1}^{T} \Big[\underbrace{-\mathbb{E}_{Q_\phi(s_t|o_{\leq t}, a_{\leq t})} \left[\log P_\theta(o_t \mid s_t, \pi)\right] - \mathbb{E}_{Q_\phi(s_t|o_{\leq t}, a_{\leq t})} \left[\log P_\theta(r_t \mid s_t, \pi)\right]}_{\text{reconstruction}} \tag{7}$$
$$+ \underbrace{\mathbb{E}_{Q_\phi(s_t|o_{\leq t}, a_{\leq t})} \left[D_{KL}(Q_\phi(s_t \mid o_t, s_{t-1}, \pi)) \parallel P_\theta(s_t \mid s_{t-1}, \pi)\right]}_{\text{dynamics}} \Big] .$$

[6] https://github.com/danijar/dreamerv2

A.2. Expected free energy for pepper

We implemented EFE using the parameterisations introduced in [14] and adapted for [25]:

- **Term 5a** was modelled as a categorical likelihood model (using normalised Dirichlet counts).
- **Term 5b** was computed as the KL divergence between the prior ($s \sim Q(s_\tau|\pi)$) and the posterior ($s \sim P(s_\tau|o_\tau, \pi)$) states. This could be computed analytically because the prior and posterior state distributions were modelled as Categorical distributions. Here, the dependency on the policy π was accounted for by using the RNN hidden state h_t summarising past actions and roll-outs.
- **Term 6a** was computed as the entropy of the observation model $P(o_\tau|s_\tau, \pi)$. Happily, the factorisation of the observation model – as independent Gaussian distributions – allowed us to calculate the entropy term in closed form.
- **Term 6b** was computed as the difference between $\log Q(s_\tau|\pi)$ and $\log P(s_\tau|D)$, where $\log Q(s_\tau|\pi)$ was approximated using a single sample from the prior model $Q(s_\tau|\theta, \pi)$. Again, the dependency on π was substituted by h_t.
- **Terms 5c and 6c** were more challenging to compute. Like [14], we rearranged the expression to $H(o_\tau|s_\tau, \theta, \pi) - H(o_\tau|s_\tau, \pi)$. This translates to $I(o_\tau; \theta|s_\tau, \pi)$, and can be approximated using Deep Ensembles [37, 55] and calculating their variance $\text{Var}_\theta[\mathbb{E}Q(o_\tau|s_\tau, \theta, \pi)]$. Here, each ensemble component can be seen as a sample from the posterior $Q(\theta|s_\tau, \pi)$. Our experiments showed that using 5 components was sufficient.

B. Experiments

FrozenLake For these experiments, we simulated the agent in five distinct situations ranging from a non-volatile, static environment to a highly volatile one i.e., a different FrozenLake map every step. For all episodes in the static setting, the agent was initialised at a fixed location with no changes to the FrozenLake map throughout that particular episode. Conversely, agents operating in the volatile setting were initialised at a different location each time. Moreover, the FrozenLake map was also changed every N steps – given the desired volatility level. For 100% volatility the map changed every step, 75% volatility corresponded to map changes every 10 steps, 50% volatility corresponded to map changes every 20 steps and 25% volatility corresponded to map changes every 40 steps. Additionally,

the generative model used for these experiments was trained in a volatile setting where the map changed every 5 steps (Table 1).

TileWorld For these experiments, we simulated the agent under two conditions (non-volatile and volatile). For the non-volatile setting, the agent was initialised at a fixed location with no changes to the TileWorld map throughout training and testing. In the volatile setting, for every K step, we toggle alternate between a randomly sampled map and the original map. This allowed us to simulate uncertain states that trigger exploratory behaviour. The generative model used for these experiments was trained in a volatile setting where the map changed every 10 steps (Table 1).

Table 1. Training parameters

Parameter	FrozenLake	TileWorld
Planning Horizon	15 steps	15 steps
Episode Length	50 steps	200 steps
Reset Every	5 steps	10 steps
No. Episodes	50 episodes	50 episodes
No. State Categories	64 categories	32 categories
No. State Dimensions	50 dimensions	50 dimensions
No. Reward Categories	4 categories	3 categories

Table 2. Preference learning parameters for long-term learning

Parameter	FrozenLake	TileWorld
Planning Horizon	15 steps	15 steps
Episode Length	50 steps	200 steps
No. Episodes	50 episodes	50 episodes
Reset Map Every	$1, 10, 20, 40, 50$ steps	10 steps
No. Agents	10 agents	3 agents

B.1. Computational requirements

Overall, our experiments required 1344 GPU hours. Each GPU was a GeForce RTX 3090.

B.2. Image reconstruction and imagined roll-outs

For apt learning of preferences, the agent's generative model must be able to accurately infer the current (and future) states of affairs. To evaluate this for our learnt generative models, we illustrate representative examples of reconstructions encoded by the pepper agents for a particular episode. Fig.6 shows the reconstructions for FrozenLake, and Fig.7 for TileWorld. The imagined roll-outs for the TileWorld environment are shown in Fig.8.

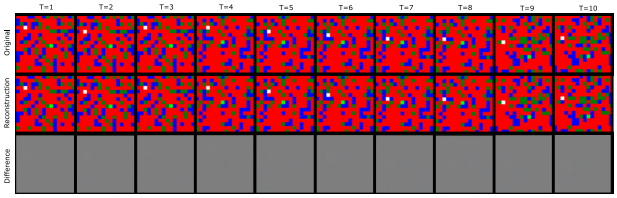

Figure 6. An example of the FrozenLake reconstruction for the first 10 steps of an episode, with map changes at $T = 4$ & 9.

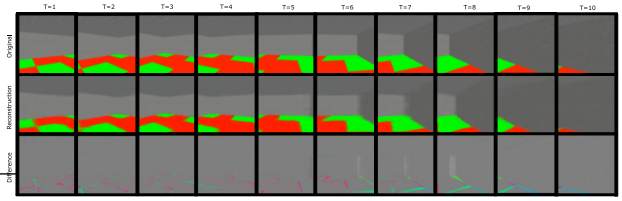

Figure 7. An example of TileWorld reconstruction for the first 10 steps of an episode.

C. Behaviour under long-term preference learning

C.1. Learnt preferences

We expected differences in learnt preferences to induce shifts in agent behaviour. For example, agents who repeatedly accrued Dirichlet pseudo-counts for the same category would exhibit preference satisfying behaviour. This would be due to high precision (or confidence) over that particular category. In contrast, an agent who accrued Dirichlet pseudo-counts for different categories would exhibit exploratory behaviour given an imprecise (or low confidence) distribution over the categories. To illustrate how different environment settings shaped preference learning we looked at the static and volatile setting where the FrozenLake map changed every step. Fig.9 shows a representative example of state preference learning under these conditions, and Fig.10 an example of reward preference learning. We observed that state preferences learnt under a static setting were precise – denoted by the repeated pseudo-count accumulation over category 25. Conversely, for the volatile setting an imprecise state preference distribution was learnt (Fig.9). Separately, we observed that the learnt reward preferences were precise – regardless of the setting. We posit that this is a consequence of differences in the reward and state category space. In other words, having a large number of state categories

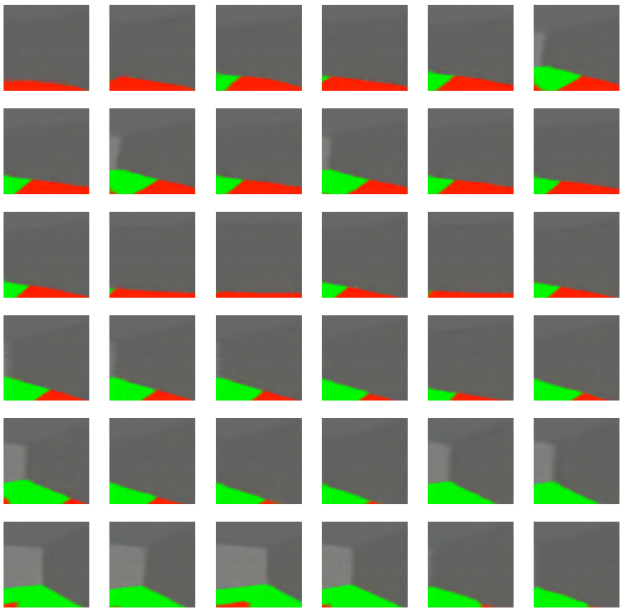

Figure 8. An example the observations from an imagined roll-out in the latent state space. These roll-outs are constructed using a random action trajectory that is propagated forward to get a latent state sequence. For each of the latent states the observations were sampled from the observation model $P(o_t|s_t)$.

allowed distinct preferences to be learnt under static and volatile settings. This is reflected in the qualitative differences seen between the two (Fig.9).

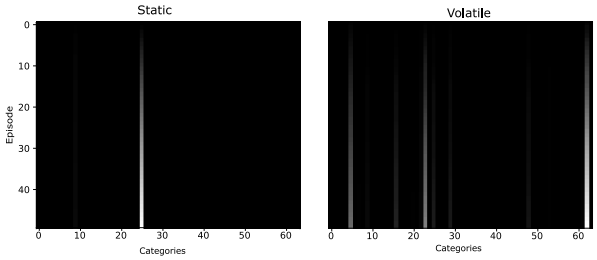

Figure 9. An example of learnt state preferences for a single agent in a static and highly volatile setting. Here, 64 state categories are presented on the x-axis and episodes on the y-axis. The first panel is for preferences learnt under a static setting, and the second for preferences learnt under a volatile setting. The scale goes from white (i.e., high Dirichlet concentration) to black (i.e., low Dirichlet concentration), and grey indicates gradations between these.

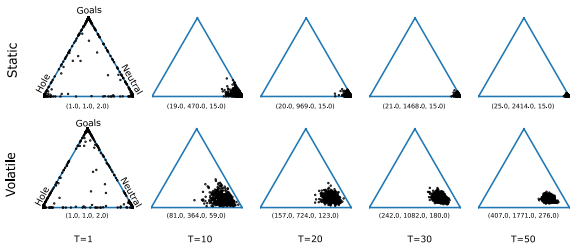

Figure 10. An example of learnt reward preferences for a single agent in a static and highly volatile setting. The first row is for preferences learnt under a static setting, and the second row for preferences learnt in a volatile setting. Each figure illustrates the Dirichlet distribution in a 3-dimensional coordinate space, i.e., 2-simplex – for a particular episode (T). Here, the concentration of dots in one corner reflect precise beliefs; and scattered dots denote imprecise beliefs. Each dot represents a single sample from the Dirichlet distribution (determined by the alpha parameters denoted at the bottom of each figure), and each plot displays 500 samples. For clarity, we collapsed Goal and Sub-goals into one category. Preferences for both static and volatile setting are initialised as uniform (i.e., $(1,1,1,1) = (1,1,2)$) denoted by the dots scattered across the simplex.

C.2. Agent trajectories

Next, we evaluated how disparate the agent trajectories were given the observed differences in preference accumulation (Figure 9 and 10). For the static setting, we observed agents satisfying their preferences by restricting movement to a small patch in the FrozenLake. This behaviour was observed consistently across all agents (i.e., different seeds) and episodes. We present a representative example in Fig.11. Separately, agents simulated in the volatile setting (where the map changed every step) learnt a bi-modal preference set (i.e., preferred to go to one of two locations in the FrozenLake). Here, the location preference depended on the initial location i.e., if the agent was initialised in a tile close to the first preferred location then it choose to go there. However, the second location was preferred if the agent was initialised close to it. We present a representative example in Fig.12. Interestingly, this behaviour was observed in agents where the environment was 100% volatile, whereas agents operating in slightly less volatile settings continued exploring (Fig.4A).

Importantly, these agents were able to disregard, noisy information about the states from the environment. To qualify this, we looked at how the variance between the posterior ($s \sim Q_\phi(s_t|h_t, o_t)$) and prior ($s \sim Q_\phi(s_t|h_t)$)

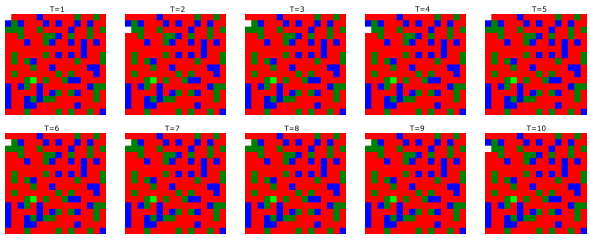

Figure 11. Representative example of the agent trajectories observed during state preference learning under a static setting.

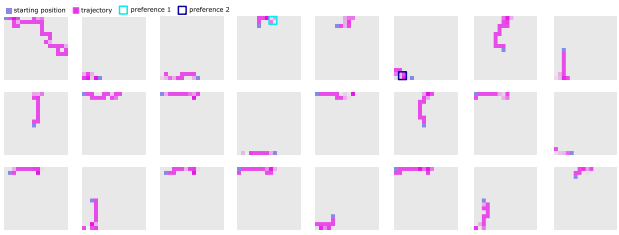

Figure 12. Representative example of the agent trajectories observed during state preference learning under the volatile setting. Each figure is an illustration of the agents trajectory for a particular episode. Here, purple is the agents starting position, pink the trajectory, cyan square denotes the first learnt preference and dark blue denotes the second learnt preference.

estimates differed across the 50 episodes for these agents (Fig.13). We observed that the posterior estimates had a greater variance across the 50 dimensions relative to the prior variance. Given how these estimates are calculated, we postulate that differences in the variance were due to the change in the FrozenLake map that the agent finds itself in after it moved one step. These high variances in the posterior estimate, under a highly volatile setting, induced a change in behaviour from exploratory to preference satisfaction.

D. Behaviour under short-term preference learning

We expected reduced preference learning timescales to influence the agent's preferred behaviour. To evaluate this, we consider a setting where the agent was equipped with a sliding preference window i.e., after k steps the previous preferences were removed in favour of new ones. To evaluate short-term preference learning, we considered state preferences with a sliding window of 5 episodes (Table 3).

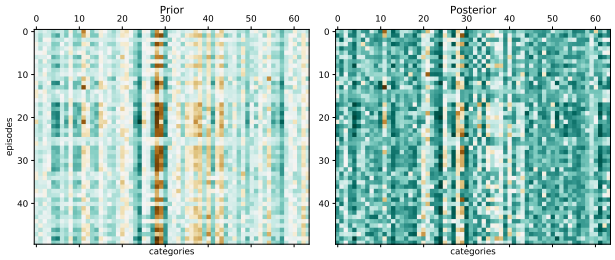

Figure 13. Variance over the estimated posterior ($s \sim Q_\phi(s_t|h_t, o_t)$) and priors ($s \sim Q_\phi(s_t|h_t)$) under the volatile setting. The scale goes from brown (low variance) to turquoise (high variance), and light shades indicate gradations between these.

Table 3. Preference learning parameters for short-term learning

Parameter	**FrozenLake**
Planning Horizon	15 steps
Episode Length	50 steps
No. Episodes	50 episodes
Reset Map Every	$1, 10, 20, 40, 50$ steps
Reset Preference Every	5 episodes
No. Agents	5 agents

For this, we looked at the preferences learnt in the static and volatile (map changes every 10 steps) setting (Fig.14). Predictably, we observed differences in the preference accumulation when the agents learnt short-term preferences regardless of the setting. Explicitly, in the static setting the accrued preferences were flexibly learnt and unlearnt over time e.g., category 24 was slowly unlearnt in favour of 11 category. Volatile conditions fostered perpetual preference uncertainty as accumulated Dirichlet pseudo-counts were repeatedly updated. Therefore, we would expect these agents to exhibit exploratory behaviour compared to agents equipped with long-term preferences due to imprecise preference learning. To quantify this behaviour, we projected the latent states onto the first two components (fitted using long-term state preferences simulation data). The short-term simulation data only mapped onto a small space in Fig.3 latent space. Furthermore, there was no clear separation between projected latent states across the the volatile and static settings. This is reflected in the increased state entropy (~ 0.5 nats), under both settings, as previously learnt preferences were removed (Fig.16).

Next, we considered how the exploration and preference satisfaction trade-off might vary when preferences

were learnt over a short time horizon. Using the Hausdorff distance, we evaluated how the volatility in the environment changed the agent's behaviour to either exploratory or satisfying preferences (Fig.16). In contrast to the long-term state preference learning setting, we see a (slight) linear association between environment volatility and preference satisfaction. This is consistent with our expectation that a slow removal of accumulated Dirichlet parameters engenders consistently exploratory agents.

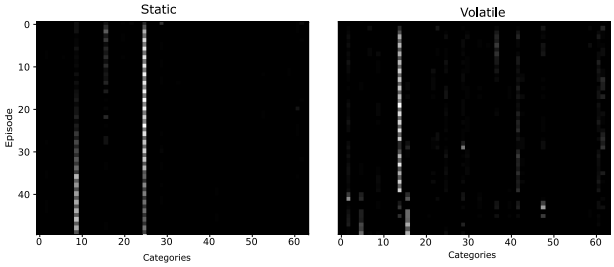

Figure 14. An example of short term state preference learning for an agent in a static and highly volatile (map change every step) environment. Here, 64 state categories are presented on the x-axis and episodes on the y-axis. The first panel is for preferences learnt under a static setting, and the second for preferences learnt in a volatile setting. The scale goes from white (high Dirichlet concentration) to black (low Dirichlet concentration), and grey indicates gradations between these.

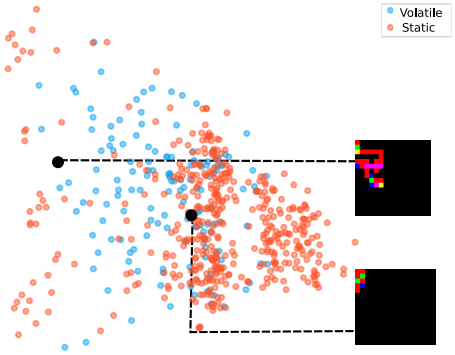

Figure 15. Visualisation of the posterior latent states (estimated using $Q_\phi(s_t|h_t, o_t)$) during state preference learning. The states have been projected onto the first two principle components (fitted using long-term state preferences simulation data), and the black circles represent their k-mean centroid. Here, the accompanying graphics present a representative agent trajectory with visited tiles highlighted from that particular cluster.

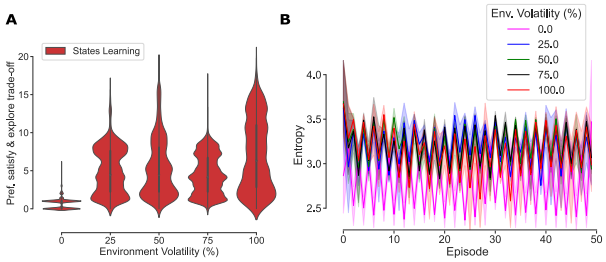

Figure 16. **A:** The violin plot illustrates the preference satisfaction and exploration trade-off measured using the Hausdorff distance [7] at different levels of volatility in the environment, when the agent had short term preferences. The x-axis denotes environment volatility: with a constant map (0%), change in map every 40 steps (25%), 20 steps (50%), 10 steps (75%) and every step (100%). The y-axis denotes the Hausdorff distance. Here, red is for the agent optimising state preference learning Eq.6. **B:** The line plot depicts the entropy over $P(s)$ across varying levels of volatility in the environment. The x-axis represents the episodes, and the y-axis entropy (in natural units). Here, the dark lines represent the mean (across 5 seeds), and shaded area the 95% confidence interval. The pink line is for 0%, blue for 25%, green for 50%, black for 75% and red for 100% volatility in the environment. The spikes in entropy correspond to overwriting of learnt preferences.