Enhanced Exploration via Variational Learned Priors

Jessica Nicholson * University of Bath
Joseph Goodier University of Bath
Akshil Patel University of Bath
Özgür Şimşek University of Bath

Abstract

Efficient exploration in reinforcement learning is challenging, especially in sparse-reward environments. Intrinsic motivation, such as rewarding state novelty, can enhance exploration. We propose an intrinsic motivation approach, called Variational Learned Priors, that uses variational state encoding to estimate novelty via the Kullback-Leibler divergence between the posterior distribution and a learned prior of a Variational Autoencoder. We assess this intrinsic reward with four different learned priors. Our results show that this method improves exploration efficiency and accelerates extrinsic reward accumulation across various domains.

1 Introduction

Exploration is a key challenge in Reinforcement Learning (RL), especially in sparse-reward environments. Intrinsic motivation, a concept rooted in psychology, enhances exploration by providing intrinsic rewards based on the agent's understanding of the environment [3, 22]. Intrinsic motivation methods include curiosity-driven exploration [19], novelty-seeking behavior [1], and count-based strategies [4, 16, 23, 24]. These methods have helped RL agents surpass human performance on the Atari-57 benchmark [2]. They are often modeled with a forward dynamics model predicting future states from current actions.

Variational Autoencoders (VAEs) [8] can model the environment by encoding states into a lower-dimensional latent space, where each state is represented by a latent variable, or latent representation, that captures its key features [11, 13]. Klissarov et al. [15] suggested using the Kullback-Leibler (KL) divergence between a VAE's fixed prior and the posterior of encoded states as an intrinsic reward. However, a fixed prior hinders the VAE's ability to adapt to the environment's evolving dynamics. Consequently, discrepancies arise in the VAE's latent space, where the prior assigns high probability density to regions that the agent has rarely visited.

We propose an intrinsic reward mechanism that uses the KL divergence between a VAE's *learned prior* and its posterior distribution, replacing the traditional fixed prior, to better encourage the exploration of novel states. Our contributions are:

1. We introduce and assess four new intrinsic rewards based on learned priors: Mixture of Gaussians prior, Generative Topographic Mapping prior, Vamp prior, and a Flow-based prior.
2. We demonstrate how the priors enhance exploration efficiency in Deepmind's DeepSea [18].
3. We show superior learning performance in MuJoCo environments (Ant, Hopper, Walker2D) compared to existing intrinsic motivation methods [6].

2 Background

A Markov Decision Process [20] consists of a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, transition dynamics P, a reward function R, and a discount factor γ. At each decision stage t, the agent observes a

*Correspondance to: jmn43@bath.ac.uk

Accepted at the Intrinsically Motivated Open-ended Learning workshop at NeurIPS 2024.

state s_t, takes an action a_t, transitions to a new state s_{t+1}, and receives an extrinsic reward r_{t+1}. In intrinsically motivated reinforcement learning, the agent receives both an intrinsic and extrinsic reward at decision stage t, using both to guide its learning.

Variational Autoencoders are a type of probabilistic generative model. The loss function consists of a reconstruction term and a regularization term, where the regularization is measured by the KL divergence between the variational posterior and the prior. The KL is a measure of the difference between one probability distribution and a reference distribution. During training this forces the VAE's posterior distribution $q(\mathbf{z}|\mathbf{x})$ to be as close as possible to the prior distribution $p(\mathbf{z})$. In a vanilla VAE, the prior $p(\mathbf{z})$ is a standard Gaussian $\mathcal{N}(0, 1)$. In this work, we employ a β-VAE [12], a modified VAE variant designed to balance reconstruction and regularization more flexibly. The β-VAE incorporates a hyperparameter β into the KL divergence term, allowing for increased control over the disentanglement of learned representations. Higher values of β emphasize the regularization term, leading to a more disentangled latent space but potentially at the cost of reconstruction quality. This adjustment enables better exploration by making the latent space representations more structured, thereby allowing the agent to identify novel states more effectively. The loss function is formalized as learning the Evidence Lower Bound Objective on the marginal likelihood of the model [14]:

$$\log p_\theta(\mathbf{x}|\mathbf{z}) \geq \mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) - \beta D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})). \tag{1}$$

Rezende and Viola [21] and Tomczak and Welling [25] argue that a fixed standard Gaussian prior may over-regularize the VAE's latent space, making it less effective at learning meaningful representations. This occurs because a fixed prior can't accurately model the aggregate posterior distribution. The aggregate posterior is the mean of all encoded samples or a mixture of the variational posteriors of all N samples [17], as defined by $q(\mathbf{z}) = \frac{1}{N} \sum_{n=1}^{N} q_\phi(\mathbf{z}|\mathbf{x}_n)$. This mismatch results in "holes" which are regions in latent space where the aggregate posterior has low probability density but the prior assigns relatively high probability density. An alternative to the standard fixed prior is to have a *learned prior*, where the aggregate posterior and the prior fit to each other. A learned prior is optimized during training, adjusting its parameters to better match the aggregate posterior. This is achieved by minimizing the KL divergence between itself and the aggregate posterior, effectively allowing the two distributions to fit each other. This dynamic adjustment enables the VAE to more accurately model complex data distributions and avoid the rigid structure imposed by a fixed prior.

3 Variational Learned Priors

We propose a new intrinsic motivation approach called Variational Learned Priors (VaLP). This uses the KL divergence between a VAE's *learned prior* and the posterior distribution of the encoded state to generate an intrinsic reward. We chose four learnable priors (see Appendix B for more details):

1. **Mixture of Gaussians (MoG)**: This prior models the aggregate posterior as a mixture of learned Gaussian distributions.
2. **Generative Topographic Modelling (GTM)**: This prior models the aggregate posterior through a transformation of a low-dimensional grid to model a mixture of Gaussian distributions.
3. **Variational Mixture of Posteriors(VaMP)**: This prior models the aggregate posterior using a mixture of Gaussian distributions with learnable pseudo-inputs.
4. **Flow-Based Density Estimator (Flow)**: This prior models the aggregate posterior by learning a series of invertible transformations from a low dimensional noise space to the latent space.

We substitute the learned prior $p(\mathbf{z})$ into the following equation to define four new intrinsic rewards:

$$r_{intrinsic} = D_{KL}\Big(q(\mathbf{z}|\mathbf{s})||p(\mathbf{z})\Big), \tag{2}$$

where $\mathbf{s}$ is the state and $\mathbf{z}$ is the encoded state. We name each intrinsic reward with a learned prior: VaLP_{MoG}, VaLP_{GTM}, $\text{VaLP}_{\text{Vamp}}$, and $\text{VaLP}_{\text{Flow}}$ (see Appendix A for the complete algorithm).

4 Experimental Results

This section presents experimental results of latent space quality, exploration efficiency, and agent performance. Environment, baselines, and evaluation metric details can be seen in appendicx C, D, E.

4.1 Improved Latent Space Quality

To evaluate the effect of different priors on a VAE's latent space, we trained VAEs with a fixed Standard prior [15] and the learnable priors $\mathrm{VaLP}_{\mathrm{MoG}}$, $\mathrm{VaLP}_{\mathrm{GTM}}$, $\mathrm{VaLP}_{\mathrm{Vamp}}$, and $\mathrm{VaLP}_{\mathrm{Flow}}$ on 6,000 states obtained from the Walker2d domain [6] using a random policy. Figure 1 shows the alignment of each prior with the encoded latent states. Alignment quality is measured by latent space coverage percentage, which estimates the proportion of posterior means falling within the top 95% of the prior density. Higher values indicate that learned priors more effectively capture the underlying data structure compared to a fixed prior.

The leftmost plot shows the rigidity of the fixed Standard prior, creating a noticeable hole between the latent states and the prior. This hole is problematic as it may result in a low KL divergence (hence low intrinsic reward) even when the observed state is very novel. The agent is incentivized to explore with high KL divergences so this latent encoding would cause the agent to receive inaccurate information, resulting in inefficient exploration. In contrast, the learned priors better align with the posterior distribution, achieving significantly higher coverage percentages, offering a more representative encoding that enhances the agent's understanding of the environment.

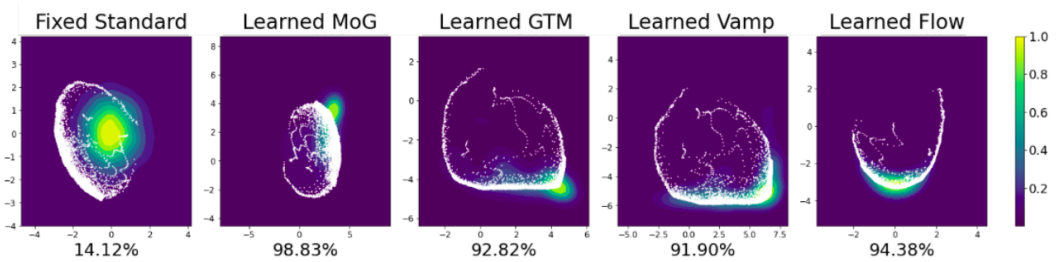

Figure 1: Two-dimensional representations of a VAE's latent space. Each VAE has been trained with a fixed Standard prior and four learned priors in the Walker2d domain. The heatmaps display the prior probability densities for each VAE, with lighter colors representing regions of higher density. White points represent the posterior means of encoded states. The Standard prior shows misalignment between the prior contours and posterior means, resulting in "holes" in the latent space. These are regions of low prior density that do not align well with the latent variables, indicating a less optimal fit to the data compared to learned priors. In contrast, the learned priors (MoG, GTM, Vamp, and Flow) demonstrate better prior-posterior alignment, as indicated by a higher coverage percentage. The coverage percentage below each plot quantifies how well the prior distribution aligns with the encoded states, with higher values suggesting a more effective latent space representation achieved by the learnable priors. Details of how this is calculated can be found in Appendix E.

4.2 Improved Exploration

Next, we evaluate the exploration efficiency of the VaLP methods against the fixed Standard prior and an agent with no intrinsic reward in the DeepSea environment [18]. Figure 2 shows the first visit to states in the 24×24 grid for each algorithm after 1000 Q-learning episodes. The heatmap colors, ranging from red (early visitation) to blue (later visitation), highlight exploration efficiency. Reaching the goal state at the lower right-hand corner is important, as it indicates successful navigation of the hardest-to-reach area in DeepSea. Figure 3 shows the state space coverage over time, representing the proportion of states visited by the agent within a fixed number of decision stages.

Figure 2 shows that the four proposed intrinsic rewards produce effective exploration, with distinct patterns of early state visitation. $\mathrm{VaLP}_{\mathrm{Flow}}$ and $\mathrm{VaLP}_{\mathrm{MoG}}$ efficiently explore the grid, reaching the goal state in the lower right-hand corner. $\mathrm{VaLP}_{\mathrm{GTM}}$ also shows strong exploration with early state visitation. With no intrinsic reward, the agent exhibits limited exploration, while $\mathrm{VaLP}_{\mathrm{Vamp}}$ and the fixed Standard prior improve exploration but fail to reach the goal.

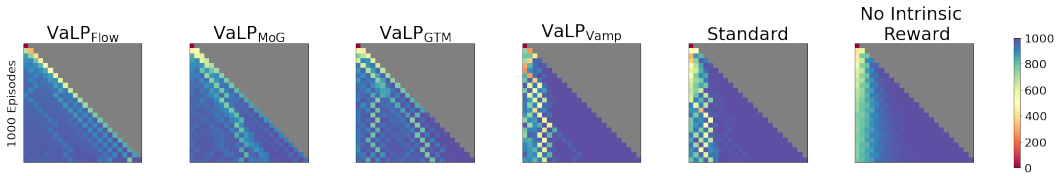

Figure 2: Heatmaps of first visit to states in 24×24 DeepSea. The coloring is linearly scaled, red indicates earlier first-time visitation and blue indicates later. The upper right section in grey is unreachable by the agent. The agent with no intrinsic reward shows very limited exploration, with early visits concentrated in a small region, far from the ideal uniform exploration across the grid. The intrinsic reward using a Standard prior shows some improvement in exploration but still does not cover the grid effectively. In contrast, the flexible priors, $\text{VaLP}_{\text{Flow}}$, VaLP_{MoG}, and VaLP_{GTM}, efficiently explore the environment and reach the goal in the lower right-hand corner. While $\text{VaLP}_{\text{Vamp}}$ also improves exploration, it, along with the Standard prior, fails to reach the goal.

In Figure 3, both the 24×24 and 48×48 environments show that $\text{VaLP}_{\text{Flow}}$ and VaLP_{MoG} demonstrate rapid convergence, with nearly 100% coverage in under 10 episodes for the smaller grid and about 90% in the larger grid. VaLP_{GTM} also performs well, converging to 100% and 80% coverage in approximately 10 and 30 episodes, respectively. $\text{VaLP}_{\text{Vamp}}$ achieves 90% coverage in the smaller grid but only 60% in the larger grid. In contrast, the no intrinsic reward and fixed Standard prior baselines exhibit slower and lower convergence, with coverage peaking at just under 80% in the smaller grid and 60% in the larger grid.

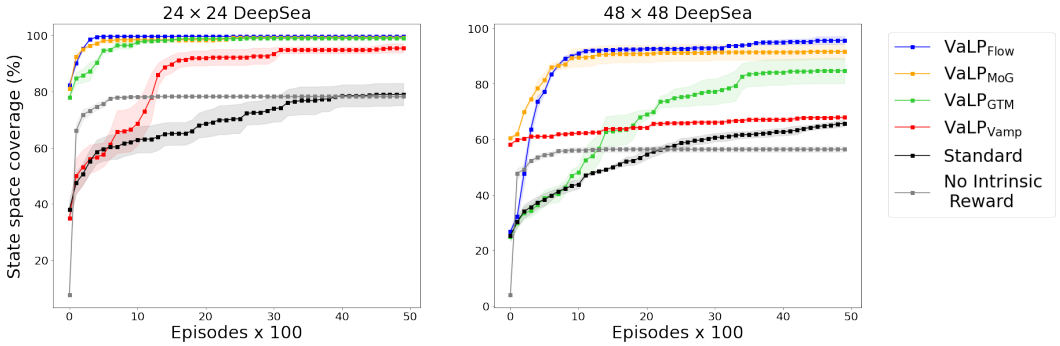

Figure 3: Percentage of state space covered. The figure shows mean values in three replications with different seeds. Each replication was run for 5000 episodes; the percentage covered was recorded every 100 episodes. The proposed VaLP methods demonstrate efficient coverage, reaching a higher coverage in fewer episodes than baselines.

4.3 Improved Agent Performance

Figure 4 presents learning curves of the VaLP methods in the MuJoCo environments [6] Walker2d, Hopper, and Ant against the fixed Standard prior [15], two intrinsic baselines RND [7], and ICM [19], and a non-intrinsic method, TD3 [10]. The VaLP methods generally improve performance across environments. VaLP matches RND and ICM in Walker2d while outperforming the fixed standard prior. In Hopper, $\text{VaLP}_{\text{Flow}}$ surpasses all baselines, while VaLP_{MoG}, $\text{VaLP}_{\text{Vamp}}$, and VaLP_{GTM} outperform ICM, TD3, and the fixed standard prior but are on par with RND. In Ant, $\text{VaLP}_{\text{Flow}}$, $\text{VaLP}_{\text{Vamp}}$, and VaLP_{GTM} outperform all baselines, although VaLP_{MoG}, while better than the fixed standard prior, still lags behind the other VaLP methods.

5 Discussion and Conclusion

In this work, we introduced a novel intrinsic motivation approach called VaLP which leverages the KL divergence between a VAE's learned prior and the posterior distribution of encoded states. Our method is designed to be plug-and-play, seamlessly integrating with existing RL frameworks. It

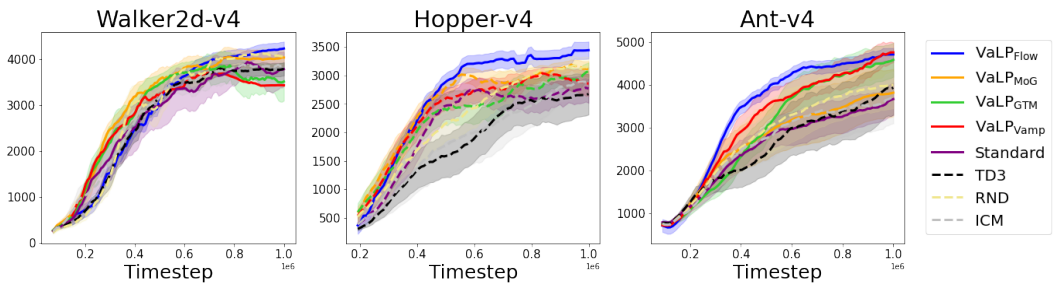

Figure 4: Learning curves for Walker2d, Hopper, and Ant-v4 environments, showing mean extrinsic rewards over time. Each curve is averaged across five seeds, with shaded areas representing the standard error of the mean. The results have been smoothed using a moving average with a window size of 10

allows for easy substitution of learned priors and provides a significant improvement over static priors, such as the fixed Standard distribution.

Our empirical evaluation across DeepSea and MuJoCo environments highlights the strengths of our approach. In the DeepSea environment, our method, specifically $\mathrm{VaLP}_{\mathrm{Flow}}$ and $\mathrm{VaLP}_{\mathrm{MoG}}$, demonstrated substantial improvements in exploration efficiency, particularly in larger state spaces. $\mathrm{VaLP}_{\mathrm{Flow}}$, in particular, exhibited exceptional performance in MuJoCo environments, likely due to its ability to rapidly learn and adapt a flexible distribution that enhances exploration efficiency. The flow model likely creates regions of high probability density in the data space, showing that it effectively reshapes the distribution to fit the data.

In conclusion, our findings underscore the advantage of having a learned prior instead of a fixed prior in a VAE's KL divergence. By adapting the reward signals based on learned distributions, our method facilitates more effective exploration in the absence of extrinsic rewards and improves sample efficiency across various tasks. Future work could explore further refinements to the learned priors, potentially incorporating additional forms of data or extending the method to new types of environments.

References

[1] Joshua Achiam and Shankar Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*, 2017.

[2] Adrià Puigdomènech Badia, Bilal Piot, Steven Kapturowski, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, and Charles Blundell. Agent57: Outperforming the atari human benchmark. In *International conference on machine learning*, pages 507–517. PMLR, 2020.

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

[4] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

[5] Christopher M Bishop, Markus Svensén, and Christopher KI Williams. Gtm: The generative topographic mapping. *Neural computation*, 10(1):215–234, 1998.

[6] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

[7] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *Seventh International Conference on Learning Representations*, pages 1–17, 2019.

[8] Yankun Chen, Jingxuan Liu, Lingyun Peng, Yiqi Wu, Yige Xu, and Zhanhao Zhang. Auto-encoding variational bayes. *Cambridge Explorations in Arts and Sciences*, 2(1), 2024.

[9] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

[10] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[11] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/2de5d16682c3c35007e4e92982f1a2ba-Paper.pdf.

[12] Irina Higgins, Loic Matthey, Arka Pal, Christopher P Burgess, Xavier Glorot, Matthew M Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR (Poster)*, 3, 2017.

[13] Irina Higgins, Arka Pal, Andrei Rusu, Loic Matthey, Christopher Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. DARLA: Improving zero-shot transfer in reinforcement learning. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1480–1490. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/higgins17a.html.

[14] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[15] Martin Klissarov, Riashat Islam, Khimya Khetarpal, and Doina Precup. Variational state encoding as intrinsic motivation in reinforcement learning. In *Task-Agnostic Reinforcement Learning Workshop at Proceedings of the International Conference on Learning Representations*, volume 15, pages 16–32, 2019.

[16] Sam Lobel, Akhil Bagaria, and George Konidaris. Flipping coins to estimate pseudocounts for exploration in reinforcement learning. In *International Conference on Machine Learning*, pages 22594–22613. PMLR, 2023.

[17] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

[18] Ian Osband, Yotam Doron, Matteo Hessel, John Aslanides, Eren Sezener, Andre Saraiva, Katrina McKinney, Tor Lattimore, Csaba Szepesvári, Satinder Singh, Benjamin Van Roy, Richard Sutton, David Silver, and Hado van Hasselt. Behaviour suite for reinforcement learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rygf-kSYwH.

[19] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[20] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[21] Danilo Jimenez Rezende and Fabio Viola. Taming vaes. *arXiv preprint arXiv:1810.00597*, 2018.

[22] Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.

[23] Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bull.*, 2(4):160–163, jul 1991. ISSN 0163-5719. doi: 10.1145/122344.122377. URL https://doi.org/10.1145/122344.122377.

[24] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

[25] Jakub Tomczak and Max Welling. Vae with a vampprior. In *International Conference on Artificial Intelligence and Statistics*, pages 1214–1223. PMLR, 2018.

A Pseudocode

Algorithm 1 Intrinsic Motivation with Different Learned Priors

Require: β-VAE encoder q_ϕ, β-VAE decoder p_ψ, policy π_θ

Let $t = 0$
Collect $D = \{s^{(i)}\}$ using random exploration policy
Pre-train β-VAE on D
Fit prior $p(z)$ to latent encodings $\{\mu_\phi(s^{(i)})\}$
for $n = 0, ..., N-1$ steps **do**
    Take action a_t, get next state s_{t+1} and extrinsic reward $r_{e_{(s_{t+1})}}$
    Compute intrinsic reward:
    $r_{i_{(s_{t+1})}} = KL(q_\phi(z|s_{t+1})||p(z))$
    Store $(s_t, a_t, s_{t+1}, r_{e_{(s_{t+1})}}, r_{i_{(s_{t+1})}})$ into replay buffer B
    if mod(t,N) == 0 **then**
        Train the Actor-Critic on return $Q(s_t, a_t) = \sum_t r_e(s_t) + \gamma Q(s_t, a_t) + \beta r_i(s_t)$
        Train the VAE on random collected states from B
    end if
end for
return solution

B Learned Priors

Mixture of Gaussians (MoG). This is a natural choice where the parameters of the mixture models are learned directly. Note the aggregate posterior: $q(\mathbf{z}) = \frac{1}{N}\sum_{n=1}^{N} q_\phi(\mathbf{z}|\mathbf{x}_n)$ is a mixture of variational posteriors, each of them Gaussian. The MoG prior is

$$p_\lambda(\mathbf{z}) = \sum_{k=1}^{K} \omega_k \mathcal{N}(\mathbf{z}|\mu_k, \sigma_k), \tag{3}$$

where K is the number of components, ω_k is a learned weighting coefficient, and $\mathcal{N}$ is a normal distribution parameterized by learnable parameters μ_k and σ_k.

Generative Topographic Mapping (GTM). Here the parameters of the mixture models are learned by transforming a low-dimensional fixed grid of K points to a higher-dimensional target domain, in our case, a Gaussian mixture model, through a transformation g_γ learned during training [5]. The GTM prior is

$$p_\lambda(\mathbf{z}) = \sum_{k=1}^{K} \omega_k \mathcal{N}(\mathbf{z}|\mu_g(\mathbf{u}_k), \sigma_k^2(\mathbf{u}_k)), \tag{4}$$

where $\mu_g(\mathbf{u}_k)$ and $\sigma_k^2(\mathbf{u}_k)$ are the outputs of the neural network g_γ. In this case, $\mathbf{u}_k$ is the fixed low-dimensional grid from which the prior is modelled, and K is the number of components in the low-dimensional grid. Again, ω_k is a learned weighting coefficient. In the GTM prior, the number of Gaussian components is equal to the number of components in the low-dimensional grid.

Variational Mixture of Posteriors (Vamp). This is a more sophisticated Gaussian mixture model that models the prior using a mixture of posterior models conditioned upon learnable pseudo-inputs in the input space [25]. The Vamp prior is

$$p_\lambda(\mathbf{z}) = \frac{1}{K}\sum_{k=1}^{K} q_\phi(\mathbf{z}|\mathbf{u}_k), \tag{5}$$

where K is the number of pseudo-inputs, $\mathbf{u}_k$ is a learnable pseudo-input with the same dimensionality as the input data. The pseudo-inputs are learned through backpropagation and can be thought of as hyperparameters of the prior, alongside parameters of the posterior. For $K << N$, the model can avoid overfitting the data.

Flow-Based Density Estimator (Flow). This prior makes no assumptions about the structure of the aggregate posterior. Called Real Non-volume preserving transformation [9], this prior is flow-based and thus learns a series of invertible transformations typically between a lower complexity distribution and a higher complexity data distribution of the same dimensionality. We use a Real-NVP to model the prior using a flow-based density without the assumption of a Gaussian Mixture model

$$p_\lambda(\mathbf{z}) = f_\lambda(\mathbf{z}), \tag{6}$$

where f_λ is an invertible flow-based neural network with learnable parameters λ.

C Environment Details

DeepSea environment comes from DeepMind's Behaviour Suite for Reinforcement Learning [18], depicted in Figure 5. This environment is an $N \times N$ grid. The agent starts in the top left corner. There is a single goal state, the grid square at the bottom right corner. Each action takes the agent to the next row but the agent can choose whether to go diagonally to the left or to the right. The episode terminates after N steps, which means that the agent can access only the lower diagonal of the grid. The agent receives a reward of 0 for moving left, $-0.01/N$ for moving right, and 1 for reaching the goal state at the bottom right corner. In this environment, we input the state to the VAE as a one-hot vector of length $N \times N$.

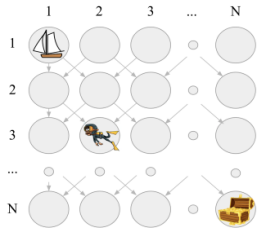

Figure 5: Example of Deep-sea exploration

MuJoCo environments Walker2d, Hopper and Ant come from OpenAI's Gymnasium [6]. They are popular benchmark domains for exploration due to their complex dynamics, continuous action space, and high-dimensional state space. The environments challenge algorithms to learn motor control and locomotion skills, making them valuable for developing robust and applicable RL methods. In these environments, we input the full observation into the VAE.

D Baseline Details

1. **Standard Prior** [15]
   Uses the standard Gaussian fixed prior within a VAE's KL-divergence to incentivize the agent to explore novel states.

$$r_{intrinsic}(S) = KL((p(Z|S)||p(Z)) \tag{7}$$

2. **RND** [7]
   Uses the prediction error of a random network as an exploration bonus aiming to reward novel states more than previously encountered ones.

$$r_{intrinsic}(S) = \|\hat{f}(s) - f(s)\|_2 \tag{8}$$

   where $f : S \to \mathbf{R}$ is a randomly initialised fixed mapping and $\hat{f}$ is trained to fit the output of f

3. **ICM** [19]
   By measuring prediction error in the latent space of an inverse dynamics model, the authors aim to measure the reducible prediction error because the latent space of the inverse dynamics model should only include information about what the agent can control.

$$r_{intrinsic}(S) = \frac{\eta}{2}\|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2 \tag{9}$$

   where $\phi(s_{t+1})$ is the feature encoding of the next state s_{t+1}, and $\hat{\phi}(s_{t+1})$ is the output of the forward model that takes a + $\phi(s)$ as input.

E Evaluation Metrics

Latent Space Quality. This will be measured by the classification accuracy of the benchmark discriminative classifiers trained on the unsupervised latent representation of the training sets of the MNIST and FashionMNIST datasets. The higher the classification accuracy on the test set of each respective dataset, the higher the quality of the learned representation.

Latent Space Coverage Percentage The coverage percentage shown below each plot is calculated by first estimating the prior density at each encoded state's mean (posterior mean) using a Gaussian KDE. Then, the proportion of posterior means that fall within the top 95% of the prior density distribution is computed. This coverage percentage indicates how well the prior distribution aligns with the latent space structure, with higher values suggesting a more effective latent space representation achieved by the learnable priors.

First visit to state. This is a record of the time when the agent visits a given state for the very first time. In the DeepSea environment, episodes are short — they terminate after N steps — so we ran the simulations for 1000 episodes. For each state, we recorded the episode number in which the state was visited for the very first time in the agent's lifetime. If the state was never visited, we assigned the state a value of 1000, the total number of episodes in the simulation.

Coverage. This is the proportion of states visited by the agent in a given (fixed) number of decision stages.

Return. This is the return obtained by the agent, calculated using only the extrinsic rewards. We present return by using learning curves.

F Hyperparameters

F.1 Latent Dimensions

To identify the ideal latent dimensions of each test environment we conducted a grid search over the following ranges:

- DeepSea: 2, 4, 6, 8, 10
- MuJoCo: 2, 4, 6, 8, 10

Experiment	Environment	Latent Dimension
DeepSea	24×24	2
	48×48	2
MuJoCo	Ant	10
	Walker2d	10
	Hopper	4
	HalfCheetah	2

Table 1: Latent dimensions for each experiment.

F.2 DeepSea

Name	Description	Value
number of episodes	Total number of episodes used to train the agent.	5000
test reward period	Frequency (in episodes) at which the agent's performance is evaluated.	100
states size	Total number of possible states.	np.prod(env.obs_space.shape)
actions size	Total number of possible actions.	env.action_space.n
hidden size	Number of neurons in the hidden layer of the neural network.	16
ICM embedding size	Dimensionality of the embedding space used by ICM.	32
LBS action size	Size of the action vector used in LBS.	1
number of values	Maximum value that can be generated for each component in the VampPrior, impacting the range of outputs for the latent representations	1
vae epochs	Training epochs for VAE.	20
latent dimension	Size of the latent space in the VAE.	See table 1
batch size	Number of samples in each batch used during VAE training.	32
image channels	Number of channels in the input images.	1
input shape	Shape of the input images to the VAE.	(3, rows, cols)
$optimizer_{intrinsic}$	Type of optimizer for the intrinsic model	Adam
$optimizer_{VAE}$	Type of optimizer for the VAE model	Adam
$\epsilon_{initial}$	Starting value for the exploration rate in the ϵ-greedy policy.	1.0
ϵ_{final}	Final value for the exploration rate in the ϵ-greedy policy.	0.1
ϵ	Fixed exploration rate.	0.1
γ	Factor used to discount future rewards.	0.9
$\alpha_{Q-learning}$	Learning rate for the Q-learning algorithm.	0.5
α_{DRND}	Scale of two intrinsic reward items.	0.9
N_{DRND}	Number of DRND target networks.	10
lr_{vae}	Learning rate for pre-training the VAE.	1e-3
M	Number of neurons in the hidden layers of the GTM prior.	256
D	Shape of the input data for the VAE, with dimensions indicating channels, height, and width of the images.	(1, rows, cols)

Table 2: Hyperparameters for the DeepSea Experiment

F.3 MuJoCo

Name	Description	Value
number of agents	How many seed repetitions to run.	3
max timesteps	Maximum time steps to run environment.	1e6
evaluation frequency	How frequent (time steps) we evaluate.	10000
evaluation episodes	How many episodes we evaluate for.	10
start timesteps	Time steps the initial random policy is used.	25000
exploration noise	Standard gaussian exploration noise.	0.1
batch size	Batch size for both actor and critic.	256
policy noise	Noise added to target policy during critic update.	0.2
noise clip	Range to clip target policy noise.	0.5
policy frequency	Frequency of delayed policy updates.	2
intrinsic weight	Weight to multiply intrinsic reward.	0.001
intrinsic update steps	How many steps to update the intrinsic reward module.	0.001
γ	Discount factor.	0.99
τ	Target network update rate.	0.005
M	Number of neurons in the hidden layers of the GTM prior.	256
D	Shape of the input data for the VAE, with dimensions indicating channels, height, and width of the images.	(1, obs_space.shape)
lr_{VAE}	Learning rate for pre-training the VAE.	1e-3
number of values	Maximum value that can be generated for each component in the VampPrior, impacting the range of outputs for the latent representations	1
image channels	Number of channels in the input images.	3
vae epochs	Training epochs for VAE.	20
latent dimensions	Size of the latent space in the VAE.	See table 1
β	Weighting term for the KL Divergence	1 for standard, 5 for learned

Table 3: Hyperparameters for the MuJoCo Experiments