$Door(s)$: Junction State Estimation for Efficient Exploration in Reinforcement Learning

Benjamin Fele, Jan Babič
Jožef Stefan Institute
Jamova cesta 39, 1000 Ljubljana, Slovenia
{benjamin.fele, jan.babic}@ijs.si

Abstract: Exploration is one of the important bottlenecks for efficient learning in reinforcement learning, especially in the presence of sparse rewards. One way to traverse the environment faster is by passing through junctions, or metaphorical doors, in the state space. We propose a novel heuristic, $Door(s)$, focused on such narrow passages that serve as pathways to a large number of other states. Our approach works by estimating the state occupancy distribution and allows computation of its entropy, which forms the basis for our measure. Its computation is more sample-efficient compared to other similar methods and robustly works over longer horizons. Our results highlight the detection of dead-end states, show increased exploration efficiency, and demonstrate that $Door(s)$ encodes specific behaviors useful for downstream learning of various robotic manipulation tasks. The code is available at https://www.github.com/benquick123/doors.

Keywords: Reinforcement learning, Intrinsic motivation, Junction States, Information theory, Curriculum learning, Exploration

1 Introduction

One of the key research problems in reinforcement learning (RL), particularly in scenarios with sparse rewards, is efficient exploration [1] and subsequent learning [2]. The core issue lies in the scarcity of feedback signals, which makes it challenging for agents to discover rewarding trajectories and learn effective policies. To address this bottleneck, various strategies have been developed to encourage exploration and accelerate learning in such environments [3].

Among existing solutions, intrinsic motivation (IM) has emerged as a prominent paradigm, providing internal reward signals to guide exploration [1, 4]. It includes novelty-based methods, which incentivize visiting less-explored states [5, 6, 7, 8], and information-theoretic approaches, which often maximize mutual information between states and actions, or states and skills [9, 10, 11, 12]. These techniques have solved hard-exploration problems in video games [7, 13, 14] and robotic control [10, 12, 15]. However, they also show drawbacks. For instance, count-based methods [5, 6, 7, 8] neglect the environment's structure, and can be inefficient compared to the utilization of additional information. Empowerment [9, 10, 16], a measure of an agent's control over the environment is theoretically capable of handling long horizons, but requires accurate modeling of long action sequences and their consequences in the environment, which remains an open problem.

To overcome these limitations, we propose a novel heuristic called $Door(s)$. Our approach identifies junction states – metaphorical doors or narrow passages in the state space – granting access to many other states. The concept of a "door" thus relates to a junction state acting as a gateway or key transition point within the state space. Example heatmaps produced by our heuristic can be seen in Figure 1a. By prioritizing traversal through these junction states, our method facilitates faster exploration of the environment, especially compared to focusing on all parts of the environment

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

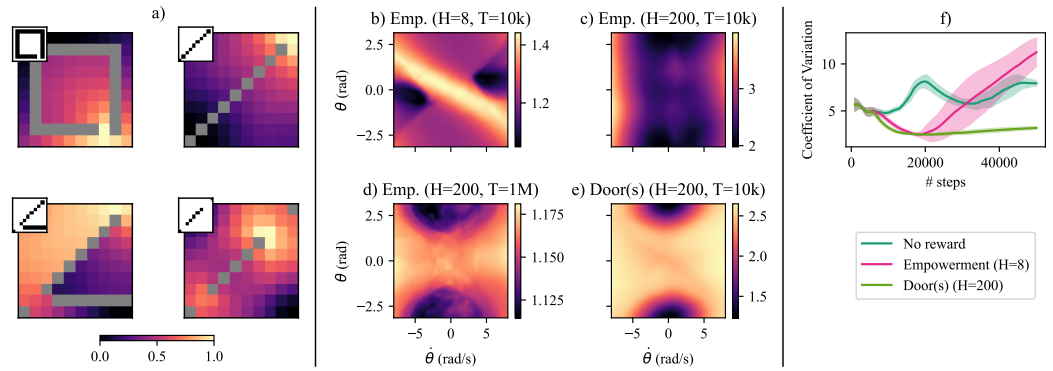

Figure 1: **(a)** Heatmaps showing the values of $Door(s)$ in various grid-world environments. Gray color represents walls through which traversal is not possible. Our junction state heuristic focuses on narrow passages between parts of the state space and prioritizes centrality when multiple are available. The plots are normalized for ease of presentation. **(b–e)** Empowerment heatmaps for the *Pendulum-v1* environment, and the result of our method. Variable H denotes horizon length, and T denotes the number of time-steps available for model training. Plots show angular velocity and angle of the pendulum on the X- and Y-axis, respectively. Our approach (e) achieves comparable results to (d) using significantly less data. **(f)** Ratio between standard deviation and mean of state visitation counts (i.e. Coefficient of Variation) with respect to the number of time-steps taken in the environment when training without, or only with heuristic rewards in the *Pendulum-v1* environment. Lower is better. Maximizing $Door(s)$ leads to greater uniformity of visited states.

without prior bias. Designed to capture the degree to which a state acts as a junction, it relates to one of the characteristics of empowerment at long horizons [17].

Unlike empowerment, which relies on potentially challenging long-horizon environment model predictions, our method simplifies the estimation process by approximating the likelihood of reaching future states without taking into account the actions taken in the process. In addition to providing a theoretical framework, we also provide implementation details enabling our approach to scale to continuous and high-dimensional state spaces. We demonstrate i) the characteristics of our junction state measure in inverse pendulum, maze, and robotic manipulation experiments; ii) showcase the role of long-horizons in junction state estimation; iii) present findings about exploration efficiency; and iv) showcase the usefulness of $Door(s)$ for mastering downstream robotic manipulation tasks.

2 Motivation

The present paper is largely motivated by practical drawbacks of empowerment [9]. As stated by Jung et al. [17], empowerment can model gateway states given a sufficiently long horizon, in addition to providing an evaluation of an agent's control from these states. However, accurately modeling states s_{t+H} resulting from the environment model $m(s_t, a_t, \ldots, a_{t+H})$, on which empowerment depends, remains an open problem. We demonstrate this in the *Pendulum-v1* environment [18] in Figure 1b–e, where it can be seen that longer horizons require a larger number of environment transitions with the state-of-the-art empowerment approximation approach [10]. We attribute the advantage of $Door(s)$ to the fact that our method works without requiring knowledge of the sequences of actions that lead to particular states. Figure 1d–e also highlights the similarity between long-horizon empowerment and $Door(s)$, although note the smaller range of the former.

Furthermore, the reasoning behind modeling junction states lies in the insight that, when used as starting points for exploration, they may lead to a larger number of diverse states in the future. This is exemplified in Figure 1f, where optimizing only over $Door(s)$ leads to greater uniformity of state visitation counts compared to empowerment or maximum entropy learning conducted with the Truncated Quantile Critics RL algorithm [19].

Even though $Door(s)$ and empowerment show some empirical similarities, our approach is tailored specifically for junction state estimation. Its results therefore differ from empowerment at small

horizons, and the lack of action dependence limits its ability to model how much control the agent has over future states. Since the heuristic is entropy-based, it is also best suited for environments with limited stochasticity.

3 Related Work

Intrinsic motivation (IM). Intrinsic motivation (IM) aims to address the limitations of purely extrinsic reward-driven RL, particularly in sparse reward environments, by providing internal reward signals that encourage exploration and skill acquisition [4]. A number of both predictive and information-theoretic approaches have been proposed in the past, where notions of learning progress, novelty, competence, and familiarity have been formalized in various ways [4].

Count-based methods aim to search the state space as uniformly as possible, rewarding unexplored areas in the process. Bellemare et al. [5] formalized this framework as part of IM, and other approaches extended the idea to novelty computation via changes in state embeddings [6, 7], or to continuous state spaces via elliptic bonuses [8]. Instead of seeking to explore the environment without bias towards specific parts of the state space, our approach prioritizes exploration that potentially leads to many other states by leveraging the knowledge about the environment structure.

Information-Theoretic IM. Information theory offers a powerful framework for formalizing IM, revisiting concepts such as surprise, novelty, and skill learning [1]. This overview focuses on variational approaches, which approximate mutual information between states and actions via lower bounds and provide a measure of agent control over environment [16, 20, 21]. Mutual information has also been optimized for skill discovery and transfer learning in [11, 12, 22, 23]. Approaches outside of this scope use an upper bound on information gain to estimate transition novelty without targeting specific regions of the state space [24], or optimize global rather than state-specific visitation entropy [25]. Unlike these methods, $Door(s)$ does not estimate mutual information; it provides a direct, policy-agnostic heuristic computing the entropy of states reachable across multiple horizons, yielding a localized signal for junction state detection. Hazan et al. [26] provide theoretical guarantees for uniform exploration but their implementation is limited to discrete states. Controllability-Aware Unsupervised Skill Discovery [27] highlights states similar to our approach in object manipulation tasks, but contrary to our work focuses on controllability of the environment.

Empowerment, an important concept related to our work, provides a principled framework for the estimation of agent's control over the environment [9]. While our approach does not exploit variational bounds, many approaches for the estimation of empowerment do [16, 20, 28]. The current state-of-the-art method for its approximation by Zhao et al. [10] estimates the capacity of the Gaussian channel and enables learning of various tasks involving stabilization, even from image data. The junction state measure we present focuses on a potentially informative signal, also captured by empowerment, which has been difficult to compute over long horizons until now.

Curriculum Learning (CL). Closely related to IM are approaches from CL that focus on accelerating training by decomposing a main task into a sequence of subtasks of increasing difficulty [2, 29]. A key aspect of CL is the design of the curriculum, which involves deciding which tasks to present to the agent and in what order [29]. Multiple ways to facilitate curricula exist, for example by progressively changing the initial or goal states, environment characteristics (e.g., object colors, number of obstacles), or reward functions [2]. Whereas many of the aforementioned works from IM provide ways of determining subtasks in the environment [9, 11, 12], other approaches from CL focus on their ordering [30, 31, 32, 33, 34]. Our approach falls into the former group, where $Door(s)$ provides a natural way of determining potentially useful behaviors, the knowledge of which can be transferred to solve another task.

4 Proposed Approach

In this section, we outline both the theoretical formulation of $Door(s)$ and an implementation that makes our approach computationally feasible.

4.1 Problem Formulation

In a Markov Decision Process $MDP = (\mathcal{S}, \mathcal{A}, r, p, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ are state and action spaces, $r(s, a)$ is a reward function, $p(s' \mid s, a)$ characterizes state transition probabilities, and $\gamma \in [0, 1)$ is a discount factor prioritizing future rewards [35], $r(s, a)$ can be formulated to promote the visitation of states that lead to the highest number of other states in a specified time-horizon H. In this paper, we introduce a heuristic reward function, $Door(s)$, that serves this purpose. The reward function is used to find a policy $\pi(s, a)$ that maximizes the cumulative return $R(s) = \sum_{t=0}^{T}[\gamma^t r(s_t, a_t) \mid s_0 = s]$. This policy can be used as a guide policy π^g in a downstream learning setting to accelerate new skill acquisition, similarly to the setup in [36].

4.2 *Door(s)* Formalization

We are interested in quantifying the dispersity of visited states s' over a time-horizon H, given the starting state s. First, we define an environment model ρ determining the t-step transition probability from state s to s':

$$\rho(s \xrightarrow{t} s') = \sum_{x \in \mathcal{S}} \left[\rho(s \xrightarrow{t-1} x) \rho(x \xrightarrow{1} s') \right]. \tag{1}$$

This is a recursive definition, where for $t = 1$, $\rho(s \xrightarrow{1} s') = p(s' \mid s)$. The latter can be derived from state transition probabilities $p(s' \mid s, a)$ by marginalizing over all actions.

Next, we define a state occupancy distribution $\Psi^{(h)}(s \rightarrow s')$ determining the fraction of time h spent in state s' whenever the trajectory starts from s:

$$\Psi^{(h)}(s \rightarrow s') = \frac{1}{h} \sum_{t=1}^{h} \left[\rho(s \xrightarrow{t} s') \right]. \tag{2}$$

Notice that the above distribution tends to the stationary distribution as $h \rightarrow \infty$ for MDPs where such a limiting distribution exists [37]. This means that $\Psi^{(h)}(s \rightarrow \cdot)$ will yield similar distributions for many or all (in the case the stationary distribution exists) starting states s when computed for large enough h. This is an undesirable characteristic, since we want to enable our junction state measure to effectively discriminate between states. We take this into account in the definition of $Door(s)$ below.

With all necessary components introduced, we now define our junction state measure. We compute the value of state s as the average entropy of Ψ over multiple horizons h:

$$Door(s) = \frac{1}{H} \sum_{h=1}^{H} \mathcal{H}^{(h)}(\mathcal{S} \mid s) = \frac{1}{H} \sum_{h=1}^{H} \left[- \sum_{s' \in \mathcal{S}} \Psi^{(h)}(s \rightarrow s') \log \Psi^{(h)}(s \rightarrow s') \right], \tag{3}$$

where $\mathcal{H}^{(h)}$ represents the entropy of $\Psi^{(h)}$ for a fixed starting state s. Variable H is a hyperparameter and denotes the maximum horizon of our measure. Intuitively, the Equation 3 yields high values for states that on average have high occupancy distribution entropies; this corresponds to visiting many states. Averaging over multiple horizons h is intended to ease the aforementioned issue of $\Psi^{(h)}$ converging to similar distributions under large h, while also capturing the changing underlying dynamics at smaller h. We further justify this in the Appendix A. The above equations work with probability distributions over discrete variables, but in the following, we propose a practical implementation working in continuous state spaces.

4.3 Implementation

The above formulation bears multiple issues that make the computation of $Door(s)$ intractable for large or continuous state spaces $\mathcal{S}$ and long horizons H. Namely, the distributions $\rho(s \xrightarrow{t} s')$ and $\Psi^{(h)}(s \rightarrow s')$ are generally unknown and have to be estimated from data, and it is not feasible to compute the entropy exactly for an arbitrary continuous probability distribution. These two points are addressed below, while some additional computational optimizations are provided in Appendix B. Appendix C highlights the quality of results from our approximation method.

4.3.1 Approximating $\Psi^{(h)}$

Instead of working with both the $\rho(s \xrightarrow{t} s')$ and $\Psi^{(h)}(s \to s')$, we simplify our approach and only approximate the latter. We do so by using a Mixture Density Network (MDN) [38], which allows us to obtain parameters of one or more multivariate Gaussian distributions approximating the data. We use three fully connected neural networks to obtain the distribution weighting parameters $\alpha \in \mathbf{R}^k_{>0}$ such that $\sum_i \alpha_i = 1$, distribution means $\mu \in \mathbf{R}^{k \times d}$, and a covariance matrix $\Sigma \in \mathbf{R}^{k \times d \times d}_{>0}$. Variable k denotes the number of components, and d denotes the dimensionality of the state space. Neural networks approximating these parameters depend both on the states s and horizon h. We indicate this dependency by writing s and h as inputs of a function, i.e., $\alpha(s, h)$, $\mu(s, h)$, and $\Sigma(s, h)$.

In line with the original MDN publication [38], we define the likelihood of being in the state s' within the horizon h when starting from the state s as:

$$\hat{\Psi}^{(h)}(s \to s') = \sum_{i=1}^{k} \left[\alpha_i(s, h) \cdot \phi(s', \mu_i(s, h), \Sigma_i(s, h))\right], \tag{4}$$

where ϕ denotes a shorthand notation for the computation of the Gaussian distribution likelihood given a sample state s' together with distribution parameters μ_i and Σ_i, and is computed in closed form [38]. The loss we minimize for fitting the neural networks approximating the MDN parameters is then:

$$\mathcal{L} = -\mathbb{E}_{(s,h,s') \sim \mathcal{D}} \left[\log \hat{\Psi}^{(h)}(s \to s')\right], \tag{5}$$

where $\mathcal{D}$ is the dataset of collected trajectories. It is important to note how exactly s, h and s' are sampled to ensure the MDN represents the state occupancy distribution Ψ as defined above. First, a trajectory of length T_E consisting of visited states is selected from the dataset. Index i of the state s is obtained by sampling $i \sim \mathcal{U}(0, T_E - 1)$, followed by obtaining the horizon $h \sim \mathcal{U}(1, \min(H, T_E - i))$, where $\mathcal{U}(\cdot, \cdot)$ represents uniform sampling from a specified interval. Finally, state s' is obtained by sampling its index uniformly from the interval $[i + 1, i + h]$. Acquiring s, h and s' this way increases sample efficiency by allowing us to reuse every trajectory multiple times and lessens the dependence on the trajectory length T_E relative to the horizon H. MDN training data is obtained through unsupervised data collection with a random exploration policy.

4.3.2 Entropy Computation

Whenever the approximated distribution is Gaussian, it allows for easy computation of its entropy [39]. However, when a mixture is involved, the entropy has to be approximated. There are various ways of estimating the entropy of a Gaussian mixture, and we find that a simple weighted average of the component entropies is sufficient for our purpose [39]:

$$\hat{\mathcal{H}}^{(h)}(\mathcal{S} \mid s) = \sum_{i=1}^{k} \left[\alpha_i(s, h) \cdot \mathcal{H}_i\right]. \tag{6}$$

Above, $\mathcal{H}_i = \frac{1}{2}\left(\log |\Sigma_i(s, h)| + d \cdot \log 2\pi e\right)$, i.e. a closed-form expression for the entropy of the i-th component of the mixture, where $|\cdot|$ denotes the matrix determinant.

5 Experiments

In this section, we present the experiments and results showcasing the characteristics of $Door(s)$ in comparison to other baselines. We set out to answer: *Which states does $Door(s)$ assign high values to?*, *What are the benefits of long horizons in junction state estimation?*, *Does optimizing $Door(s)$ lead to increased exploration?*, and *Can a prior bias towards junction states accelerate learning of a downstream task?* We conduct our experiments using multiple environments and tasks, and compare our work to various baselines described below. More implementation details and hyperparameters are available in Appendix D.

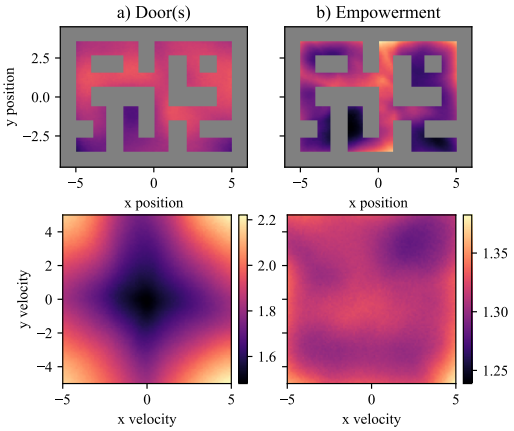

Figure 2: Comparison between $Door(s)$ and empowerment for horizon $H = 500$ in terms of position (top) and velocity (bottom). Our approach prioritizes movement over standing still and assigns low values to dead-ends in the maze. Empowerment, on the other hand, also recognizes dead-ends but inconsistently assigns high values to some corner states.

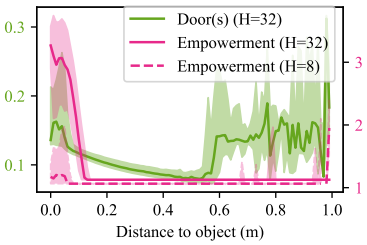

Figure 3: Relationship between distance to the object and heuristic values in the *FetchPickAndPlace-v4* environment. Plotted are medians, with shaded areas showing the 25-th and 75-th percentiles. Higher $Door(s)$ values at larger distances correspond to throwing the object. Our junction state measure captures this, while empowerment remains low. Short-horizon empowerment has the weakest relation between its value and the distance to the object.

5.1 Comparison of the Reward Landscapes

Experimental setup. We evaluate $Door(s)$ state valuation in two environments and compare it to empowerment. *PointMaze_Large-v3* is an environment consisting of a maze, where the goal is to move a point mass to various positions [40]. The agent can, given the information about the position and velocity of the point mass, exert a force on it to perform a movement. We compare the reward heatmaps using horizon $H = 500$ in *PointMaze_Large-v3*, where the total episode length is 800 time-steps. *Fetch* environments, on the other hand, entail robotic manipulation tasks involving a robotic arm and an object [40]. The agent controls the position and openness of the end-effector, and receives robotic arm and object positions and velocities, in addition to the desired position of the object. These environments feature higher, in total 30-dimensional, state spaces. In this section, we only report results for *FetchPickAndPlace-v4*, but the findings are also representative of *FetchPush-v4* and *FetchSlide-v4*. Episode length in this environment is 50 time-steps. We use the environment with shorter episodes and horizon lengths to showcase the benefit of measuring junction states and ensuring that the empowerment can reliably converge.

We use the empowerment implementation from Zhao et al. [10] and, as suggested in the original publication, use an action encoder to ease training over long horizons. In *FetchPickAndPlace-v4*, we train empowerment over an 8-step horizon, in addition to providing comparisons for $H = 32$. We train the heuristic models in both environments with experience collected over 1M time-steps.

Results. Qualitative results of the heuristics for the *PointMaze_Large-v3* in Figure 2 show convergence to a reward landscape that rewards high velocities and central positions near maze intersections. This is in contrast with empowerment, which shows inconsistent results. It does not prioritize any specific speed of movement, and while it assigns low values to dead-ends, it also unevenly rewards some maze corners, while not others. We attribute these findings to the problems associated with long-horizon training, whereas our method scales successfully.

We use Figure 3 to demonstrate the importance of horizon length for junction state estimation. In this scenario, we plot the relation between the value of the heuristic and the distance between the end-effector position. This is the only manipulable part of the environment, so the state values change more in the object proximity as expected. Figure 3 presents two important experimental results. First, short horizons are less suitable for capturing junction states due to the values only changing in direct proximity to the object in addition to exhibiting a smaller range. Second, computing the degree to which a state is a gateway at $H = 32$ results in focusing on interaction of the arm with the object for both $Door(s)$ and empowerment, while only $Door(s)$ captures the potential interest-

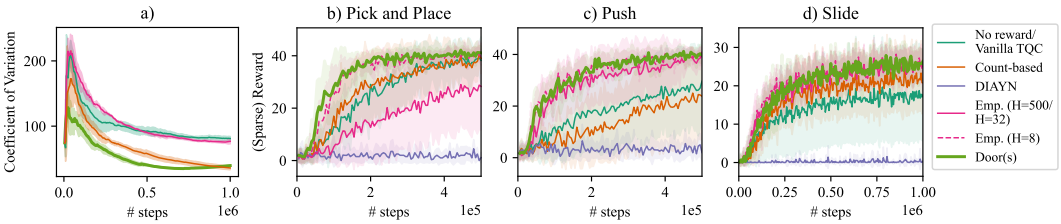

Figure 4: **(a)** Comparisons between $Door(s)$ and multiple baselines in terms of mean coefficient of variation (solid lines) with standard deviations (shaded areas) in the *PointMazeLarge-v3* environment. For smaller number of time-steps, optimizing $Door(s)$ leads to larger diversity of visited states, but count-based exploration catches up towards the end of training due to a dynamic reward. **(b-d)** Initializing the policy with the agent pretrained to maximize the value of $Door(s)$ significantly reduces the mean number of steps to convergence in multiple tasks. Empowerment is similar, while other methods are notably worse.

ingness of states at longer distances. These entail throwing or dropping an object – something that empowerment, which prioritizes control, does not capture.

5.1.1 State Visitation Distribution

Experimental setup. These experiments are conducted only in the *PointMaze_Large-v3* environment due to its low state-space dimensionality, which enables computation of the diversity of visited states. We use heuristics learned in the previous section with horizon $H = 500$ and train an agent to maximize them using the Truncated Quantile Critics (TQC) algorithm [19] over 1M steps. TQC is a maximum entropy method with a built-in action entropy term that encourages exploration. Alongside $Door(s)$ and empowerment, we include a count-based baseline [5] encouraging non-biased environment exploration. Results are averaged over 5 random seeds.

Results. The results from the Figure 1f in the *Pendulum-v1* environment are validated in Figure 4a: with $Door(s)$, the agent starts by exploring the most states as indicated by the coefficient of variation of state counts [41]. In this environment, optimizing empowerment provides a similar incentive for exploration to maximum entropy learning with no additional reward. Count-based exploration continues to increase the diversity of visited states due to its propensity for uniform exploration towards the end of training, but our method shows a clear advantage at the beginning.

5.1.2 Using *Door(s)* for pretraining

Experimental setup. These experiments are conducted in the *Fetch* environments [40], where the goal is to pick and place, push, or slide an object to a target position. The difference between pushing and sliding is that the latter requires moving a puck beyond the arm's reach, while picking and placing requires grasping. These environments naturally involve gateway states at points of object interaction. In a similar setup to the previous section, we first train the agent to maximize only the heuristic reward, and then use transfer learning to a downstream task. Transfer learning is done by warm-starting the downstream policy with a pretrained one and immediately collecting trajectories under that policy. Agents are trained for 500k (*PickAndPlace-v4*, *Push-v4*) or 1M (*Slide-v4*) steps. We compare our method against a sparse-reward baseline without pretraining, and agents pretrained on short- ($H = 8$) and long-horizon ($H = 32$) empowerment [10], a count-based approach [5], and DIAYN [11]. In the latter, the pretrained discriminator evaluates behavior discriminability in a given state, providing a potentially informative reward.

Additionally, we also quantitatively evaluate convergence. We log the number of environment steps required to reach 95% of the maximum or minimum of the monitored performance metrics. During heuristic model learning, we monitor the test set losses, while in pretraining and downstream learning we monitor the heuristic and main episodic rewards, respectively. We smooth the values using the exponential moving average. We use 5 random seeds in both stages of pretraining and 10 random seeds in downstream learning.

Table 1: Sample efficiency in terms of the number of steps to convergence of various methods. Both stages of pretraining are conducted on the *FetchPickAndPlace-v4* task. We report the mean number of steps with standard deviations ($\times 10^5$). The values in parentheses indicate the number of successful runs. DIAYN converges very quickly during pretraining, followed by $Door(s)$. The latter also provides consistent results in downstream learning, whereas DIAYN does not.

Method	Heuristic learning	Pretraining w/ heuristic	Downstream learning		
			Pick-and-place	*Push*	*Slide*
Vanilla TQC	/	/	2.52 ± 0.62 $(\underline{9/10})$	2.35 ± 1.14 $(6/10)$	3.91 ± 1.79 $(7/10)$
Count-based	/	1.56 ± 2.03	1.97 ± 0.54 $(\underline{9/10})$	3.12 ± 1.03 $(5/10)$	4.06 ± 1.29 $(\underline{9/10})$
DIAYN	$\mathbf{0.86 \pm 0.12}$	$\mathbf{0.11 \pm 0.03}$	$n/a \pm n/a$ $(0/10)$	$n/a \pm n/a$ $(0/10)$	$n/a \pm n/a$ $(0/10)$
Emp. ($H = 32$)	9.23 ± 0.16	1.42 ± 0.22	2.73 ± 0.95 $(6/10)$	$\mathbf{1.64 \pm 0.48}$ $(\underline{9/10})$	3.65 ± 1.05 $\mathbf{(10/10)}$
Emp. ($H = 8$)	2.39 ± 0.22	2.36 ± 1.35	$\underline{1.45 \pm 0.31}$ $\mathbf{(10/10)}$	1.80 ± 0.30 $\mathbf{(10/10)}$	$\mathbf{3.24 \pm 0.91}$ $\mathbf{(10/10)}$
$Door(s)$ ($H = 32$)	$\underline{1.19 \pm 0.24}$	$\underline{1.26 \pm 0.51}$	$\mathbf{1.35 \pm 0.27}$ $\mathbf{(10/10)}$	$\underline{1.74 \pm 0.36}$ $\mathbf{(10/10)}$	$\underline{3.38 \pm 1.09}$ $\mathbf{(10/10)}$

Results. In general, the results from Figure 4b–d can be split into two groups: $Door(s)$ and empowerment help with downstream learning due to a useful pretrained policy, while count-based and DIAYN-based pretraining lead to the same or worse results as with random initialization (Vanilla TQC). Short- and long-horizon empowerment fare similarly, since the main difference between them is in the pretraining phases. Notice the consistency of results for our method, whereas using Vanilla TQC and long-horizon empowerment exhibits relatively larger variances. DIAYN fails to converge in all experimental runs due to its heuristic reward promoting states away from the object.

Among all baseline methods, Table 1 shows that DIAYN converges fastest in heuristic learning and pretraining but collapses to trivial behavior. $Door(s)$ is the only method needing little data in both pretraining stages while staying consistent in downstream learning. Empowerment requires much more data during heuristic model training, and the main difference between short- and long-horizons in the second stage of pretraining is that the former takes more steps to converge – likely due to weaker reward signals around junction states. This highlights the importance of long horizons in junction state optimization. $Door(s)$ is the most consistent in downstream learning, succeeding in all experimental runs with the fewest or second-fewest steps to converge. For fair comparison, we report means and standard deviations of steps to convergence for successful runs only.

Even though training from scratch is faster than transfer learning with both pretraining stages, our method and some other baselines enable reusable task-agnostic heuristic rewards and pretrained policies, as long as the environment dynamics stay the same. Following [11] we thus treat the pretraining steps as "free". The three training stages could, however, be merged into a single run for better sample efficiency, which we leave for future work.

6 Conclusion

This paper introduces $Door(s)$, a heuristic for identifying junction states leading to diverse future outcomes without relying on action-conditioned models. The method is efficient to compute, works well with long horizons, and scales to continuous state spaces thanks to the Mixture Density Networks enabling robust training and fast entropy estimation. Empirical results show that $Door(s)$ improves exploration, helps avoid dead-ends, and speeds up learning of downstream tasks. These findings suggest $Door(s)$ as a practical tool for structure-aware environment exploration in RL, with potential to extend beyond purely robotic applications.

7 Limitations

Even though the presented approach offers many advantages and introduces a useful heuristic for reinforcement learning, it is not without shortcomings. Conceptually, $Door(s)$ is designed in a way that does not allow reliable estimation of junction states in stochastic environments, especially when the stochasticity is not uniform across the state space. Since our approach attributes high values to high-entropy states, it is prone to the "noisy-TV problem" [1]. A potential solution would be the explicit detection of such regions and their incorporation into the heuristic measure.

The second set of limitations stems from the method used for estimating $Door(s)$. For instance, when behaviors in the state space exhibit discontinuities (e.g., teleporting the agent), such diversity is not accurately captured. While increasing the number of Gaussian components can help fit the data, a more general solution would be preferred. However, entropy estimation poses a computational bottleneck that makes this effort challenging. Encoding states into latent representations might help address this issue. Additionally, our framework currently permits only offline estimation of $Door(s)$ and assumes a uniform action distribution, requiring further interaction with the environment. In the future, a scheme similar to [10] could be used to improve efficiency, which could be even further extended by training a state transition model to reduce the number of simulated steps.

This paper is only limited to a small number of robotic tasks, and not all reinforcement learning problems will necessarily benefit from exploiting junction states. We have tested up to 30-dimensional state spaces, but the present results are unclear about what the upper state-dimensionality limit is. However, once the number of dimensions grows too high, a potential workaround could involve learning a latent representation, and feed the latent vectors to the MDN as opposed to the raw state values. While applying $Door(s)$ learned in the simulation to a real-world setup should be relatively straight-forward given the environment dynamics remain similar, training the heuristic in the real-world from scratch would have to take into consideration the drawbacks described in previous two paragraphs.

Acknowledgments

This work was supported by the Slovenian Research Agency (research core funding) under Grant P2-0076. We also thank the reviewers for their constructive feedback.

References

[1] A. Aubret, L. Matignon, and S. Hassas. An Information-Theoretic Perspective on Intrinsic Motivation in Reinforcement Learning: A Survey. *Entropy*, 25(2):327, Feb. 2023. ISSN 1099-4300. doi:10.3390/e25020327.

[2] R. Portelas, C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer. Automatic curriculum learning for deep RL: A short survey. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20, pages 4819–4825, Yokohama, Yokohama, Japan, Jan. 2021. ISBN 978-0-9992411-6-5.

[3] T. Yang, H. Tang, C. Bai, J. Liu, J. Hao, Z. Meng, P. Liu, and Z. Wang. Exploration in Deep Reinforcement Learning: A Comprehensive Survey, July 2022.

[4] P.-Y. Oudeyer and F. Kaplan. What is intrinsic motivation? A typology of computational approaches. *Frontiers in Neurorobotics*, 1, Nov. 2007. ISSN 1662-5218. doi:10.3389/neuro.12.006.2007.

[5] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying Count-Based Exploration and Intrinsic Motivation. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[6] R. Raileanu and T. Rocktäschel. RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments. In *International Conference on Learning Representations*, Sept. 2019.

[7] T. Zhang, H. Xu, X. Wang, Y. Wu, K. Keutzer, J. E. Gonzalez, and Y. Tian. NovelD: A Simple yet Effective Exploration Criterion. In *Advances in Neural Information Processing Systems*, volume 34, pages 25217–25230. Curran Associates, Inc., 2021.

[8] M. Henaff, R. Raileanu, M. Jiang, and T. Rocktäschel. Exploration via Elliptical Episodic Bonuses. *Advances in Neural Information Processing Systems*, 35:37631–37646, Dec. 2022.

[9] A. Klyubin, D. Polani, and C. Nehaniv. Empowerment: A universal agent-centric measure of control. In *2005 IEEE Congress on Evolutionary Computation*, volume 1, pages 128–135 Vol.1, Sept. 2005. doi:10.1109/CEC.2005.1554676.

[10] R. Zhao, K. Lu, P. Abbeel, and S. Tiomkin. Efficient Empowerment Estimation for Unsupervised Stabilization. In *International Conference on Learning Representations*, Oct. 2020.

[11] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is All You Need: Learning Skills without a Reward Function. In *International Conference on Learning Representations*, 2019.

[12] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman. Dynamics-Aware Unsupervised Discovery of Skills. In *International Conference on Learning Representations*, Sept. 2019.

[13] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven Exploration by Self-supervised Prediction. In *Proceedings of the 34th International Conference on Machine Learning*, pages 2778–2787. PMLR, July 2017.

[14] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. In *7th International Conference on Learning Representations (ICLR 2019)*, pages 1–17, May 2019.

[15] S. Blaes, M. Vlastelica Pogančić, J. Zhu, and G. Martius. Control What You Can: Intrinsically Motivated Task-Planning Agent. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[16] M. Karl, P. Becker-Ehmck, M. Soelch, D. Benbouzid, P. van der Smagt, and J. Bayer. Unsupervised Real-Time Control Through Variational Empowerment. In T. Asfour, E. Yoshida, J. Park, H. Christensen, and O. Khatib, editors, *Robotics Research*, pages 158–173, Cham, 2022. Springer International Publishing. ISBN 978-3-030-95459-8. doi:10.1007/978-3-030-95459-8_10.

[17] T. Jung, D. Polani, and P. Stone. Empowerment for continuous agent—environment systems. *Adaptive Behavior*, 19(1):16–39, Feb. 2011. ISSN 1059-7123. doi:10.1177/1059712310392389.

[18] F. Foundation. Gymnasium. https://github.com/Farama-Foundation/Gymnasium, Feb. 2023.

[19] A. Kuznetsov, P. Shvechikov, A. Grishin, and D. Vetrov. Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics. In *Proceedings of the 37th International Conference on Machine Learning*, pages 5556–5566. PMLR, Nov. 2020.

[20] S. Mohamed and D. Jimenez Rezende. Variational Information Maximisation for Intrinsically Motivated Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[21] R. Houthooft, X. Chen, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. VIME: Variational Information Maximizing Exploration. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[22] V. Campos, A. Trott, C. Xiong, R. Socher, X. Giro-I-Nieto, and J. Torres. Explore, Discover and Learn: Unsupervised Discovery of State-Covering Skills. In *Proceedings of the 37th International Conference on Machine Learning*, pages 1317–1327. PMLR, Nov. 2020.

[23] D. Warde-Farley, T. V. de Wiele, T. Kulkarni, C. Ionescu, S. Hansen, and V. Mnih. Unsupervised Control Through Non-Parametric Discriminative Rewards. In *International Conference on Learning Representations*, Sept. 2018.

[24] B. Sukhija, S. Coros, A. Krause, P. Abbeel, and C. Sferrazza. MaxInfoRL: Boosting exploration in reinforcement learning through information gain maximization. In *The Thirteenth International Conference on Learning Representations*, Oct. 2024.

[25] M. Mutti, L. Pratissoli, and M. Restelli. Task-Agnostic Exploration via Policy Gradient of a Non-Parametric State Entropy Estimate. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):9028–9036, May 2021. ISSN 2374-3468. doi:10.1609/aaai.v35i10.17091.

[26] E. Hazan, S. Kakade, K. Singh, and A. V. Soest. Provably Efficient Maximum Entropy Exploration. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2681–2691. PMLR, May 2019.

[27] S. Park, K. Lee, Y. Lee, and P. Abbeel. Controllability-Aware Unsupervised Skill Discovery. In *Proceedings of the 40th International Conference on Machine Learning*, pages 27225–27245. PMLR, July 2023.

[28] J. Choi, A. Sharma, H. Lee, S. Levine, and S. S. Gu. Variational Empowerment as Representation Learning for Goal-Conditioned Reinforcement Learning. In *Proceedings of the 38th International Conference on Machine Learning*, pages 1953–1963. PMLR, July 2021.

[29] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone. Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey. *Journal of Machine Learning Research*, 21(181):1–50, 2020. ISSN 1533-7928.

[30] C. Colas, P. Fournier, M. Chetouani, O. Sigaud, and P.-Y. Oudeyer. CURIOUS: Intrinsically Motivated Modular Multi-Goal Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 1331–1340. PMLR, May 2019.

[31] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Wiele, V. Mnih, N. Heess, and J. T. Springenberg. Learning by Playing Solving Sparse Reward Tasks from Scratch. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4344–4353. PMLR, July 2018.

[32] R. Portelas, C. Colas, K. Hofmann, and P.-Y. Oudeyer. Teacher algorithms for curriculum learning of Deep RL in continuously parameterized environments. In *Proceedings of the Conference on Robot Learning*, pages 835–853. PMLR, May 2020.

[33] S. Racanière, A. K. Lampinen, A. Santoro, D. P. Reichert, V. Firoiu, and T. P. Lillicrap. Automated curricula through setter-solver interactions. *CoRR*, abs/1909.12892, 2019.

[34] P. Klink, H. Yang, C. D'Eramo, J. Peters, and J. Pajarinen. Curriculum Reinforcement Learning via Constrained Optimal Transport. In *Proceedings of the 39th International Conference on Machine Learning*, pages 11341–11358. PMLR, June 2022.

[35] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT press, 2018.

[36] I. Uchendu, T. Xiao, Y. Lu, B. Zhu, M. Yan, J. Simon, M. Bennice, C. Fu, C. Ma, J. Jiao, S. Levine, and K. Hausman. Jump-Start Reinforcement Learning. In *Proceedings of the 40th International Conference on Machine Learning*, pages 34556–34583. PMLR, July 2023.

[37] E. Neamat. *Stationary Distribution of Markov Chain*. 2023.

[38] C. M. Bishop. Mixture density networks. *Mixture density networks*, 1994. ISSN NCRG/94/004.

[39] A. Kolchinsky and B. D. Tracey. Estimating Mixture Entropy with Pairwise Distances. *Entropy*, 19(7):361, July 2017. ISSN 1099-4300. doi:10.3390/e19070361.

[40] R. de Lazcano, K. Andreas, J. J. Tai, S. R. Lee, and J. Terry. Gymnasium robotics, 2024.

[41] H. Abdi. Coefficient of variation. *Encyclopedia of research design*, 1(5):169–171, 2010.

[42] D. Gilboa, A. Pakman, and T. Vatter. Marginalizable Density Models, June 2021.

[43] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba. Hindsight Experience Replay. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Appendix

A Study on querying one vs. multiple *h*

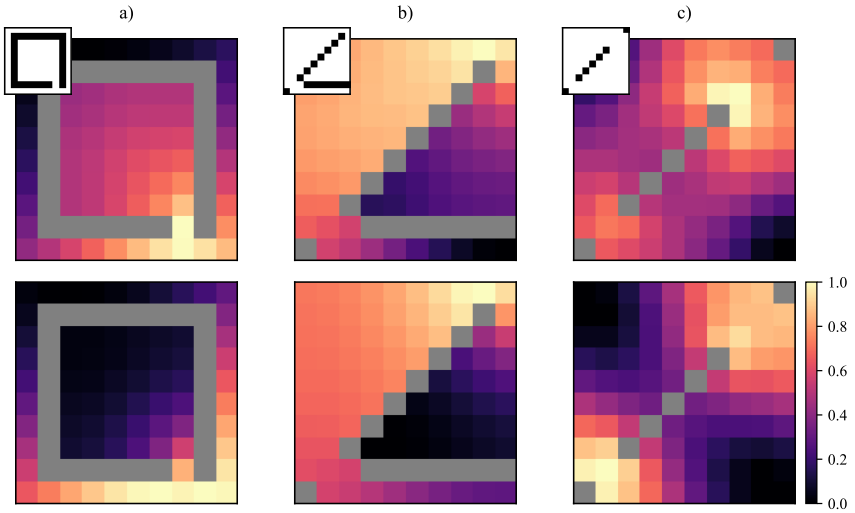

Figure 5: Differences in produced heatmaps in various grid-world environments. Upper row shows results with querying multiple horizons, while the bottom row shows querying only $h = H$. Multiple queries lead to accurate maximum state values at junctions (a), detect dead-ends (b), provide denser reward gradients (a-b), and prioritize centrality (c).

One of the design decisions in our methodology is the computation of entropy over multiple horizons $h \in [1, H]$, instead of querying state entropy only at $h = H$. We justify this choice by noting that multiple horizons help capture changing dynamics across different time-scales. An ablation study in the grid-world environment (Figure 5) illustrates this effect. Only multiple queries successfully identify junctions in the environment while also providing a denser heuristic value distribution (Figure 5a). Querying multiple horizons also captures dead-ends (Figure 5b) and focuses on central states (Figure 5c). These results suggest that using not just a single (long) horizon, but the full range, is key to identifying gateway states in the environment.

B Additional *Door(s)* optimizations

In addition to the computational optimizations presented in Subsection 4.3, we also implement several other minor improvements. First, we do not use average entropy values directly as initially

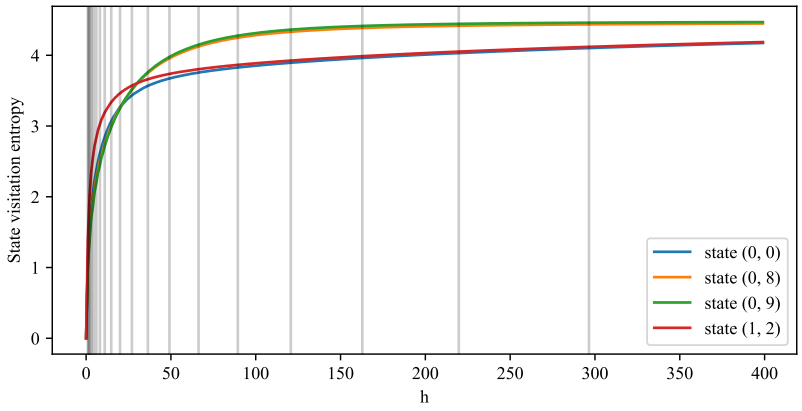

Figure 6: Example of entropies with respect to the horizons h for four states in a toy grid-world environment. Colored lines denote different squares on the grid. The values change more at the beginning, but converge to similar values. Instead of querying the model for every horizon length h, we instead conduct queries at $\dot{h}$ as denoted by gray vertical lines.

described, but instead shift and scale them to obtain comparable values across state spaces of different dimensionalities, and second, computing entropy for all $h \in [1, H]$ can become a computational bottleneck, which we also mitigate.

B.1 Ensuring Similar Scaling and Positive Values

The entropy of the Gaussian distribution depends on the dimensionality d of the state space, so we first scale the result from Equation 6 by computing $\hat{\mathcal{H}}^{(h)}(\mathcal{S} \mid s)/d$ to obtain entropy per dimension. This operation is intended to reduce the variation in reward range across different environments and improve hyperparameter robustness. Following this, since differential entropy can be negative, we ensure $\hat{\mathcal{H}}^{(h)}(\mathcal{S} \mid s) > 0$ by:

$$\hat{\mathcal{H}}^{(h)}_{>0}(\mathcal{S} \mid s) = \begin{cases} e^{\hat{\mathcal{H}}^{(h)}(\mathcal{S}|s)}, & \text{if } \hat{\mathcal{H}}^{(h)}(\mathcal{S} \mid s) < 0 \\ \hat{\mathcal{H}}^{(h)}(\mathcal{S} \mid s) + 1, & \text{otherwise} \end{cases}. \tag{7}$$

The last step is not strictly necessary but can be useful – for example, when additional dynamic weighting is applied to the value of $Door(s)$.

B.2 Reducing the Number of Entropy Computations

During our preliminary experiments, we observed that entropy changes logarithmically with respect to the horizon length h. This is shown in Figure 6; the entropy for the states presented in the graph changes rapidly at short horizons but approaches similar values logarithmically at longer horizons.

Since querying the MDN for every horizon $h \in [1, 2, ..., H]$ leads to redundant computations – assuming that entropies at longer horizons are less informative – we instead space the queries exponentially. Suppose we want to query the i-th horizon, where $i \in [1, \dot{H}]$. We compute its value as $\dot{h}_i = \sqrt[\dot{H}]{H^{i-1}}$, where $\dot{H}$ is the total number of queries we perform. The result of this operation is shown as vertical lines in Figure 6. Additionally, we scale all horizons h and $\dot{h}$ to the interval $[\frac{1}{H}, 1]$ to simplify prediction with neural networks during both training and inference.

In order to obtain an accurate approximation of $Door(s)$ from Equation 3, each resulting entropy $\hat{\mathcal{H}}^{(\dot{h}_i)}$ must be appropriately weighted based on the gap between two consecutive horizons $\dot{h}_i$ and $\dot{h}_{i-1}$:

$$\hat{Door}(s) = \frac{1}{2H} \sum_{i=1}^{\dot{H}} \left[(\dot{h}_i - \dot{h}_{i-1}) \cdot \hat{\mathcal{H}}^{(\dot{h}_i)}(\mathcal{S} \mid s) \right], \tag{8}$$

where $\dot{h}_0 = 0$.

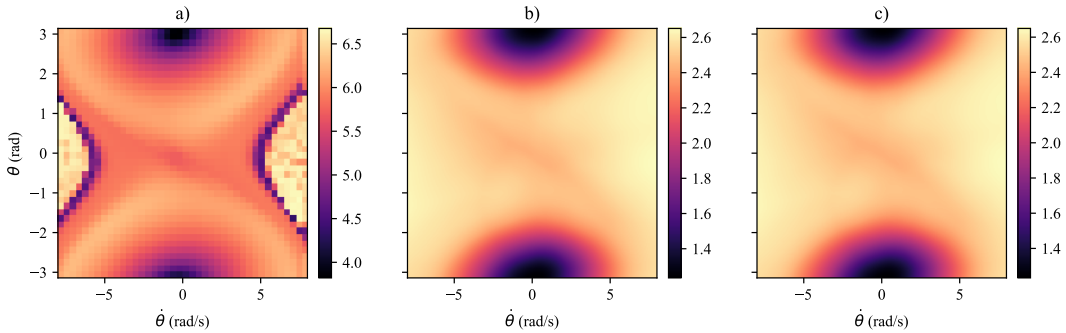

Figure 7: Differences in $Door(s)$ estimation with discretization of the state space (a), querying all horizons (b), and logarithmically spaced horizons (c) for the continuous case approximated using a MDN. Approximating with a MDN loses some detail in states with high angular velocity compared to the discrete case, but captures the drop in values at the downright and upright positions. Querying the MDN with logarithmically spaced horizons (c) shows no intelligible difference compared to (b).

C *Door(s)* approximation study

Unless the environment state transition model is known, $Door(s)$ must be approximated. In this section, we compare approximating $Door(s)$ by discretizing the state space to obtaining the state occupancy distribution $\Psi^{(h)}$, versus approximating $\Psi^{(h)}$ using a Mixture Density Network (MDN). Figure 7a–b shows the similarities and differences between the two approaches, presented with heatmaps from the *Pendulum-v1* environment. The main qualitative difference appears in the low-value boundary captured by the discrete approximation when the pendulum has a large angular velocity. Note that the discrete approximation is implemented using a tabular representation, which requires substantially more data to produce reliable visualizations and offers lower query resolution compared to the continuous case.

The computational optimization presented in Appendix B.2, exemplified by the heatmaps in Figure 7b–c, shows no intelligible difference in heuristic values despite a 10-fold decrease in reward model queries between the non-optimized (b) and optimized case (c).

D Training Details and Experiment Hyperparameters

We split this section into the environment, heuristic and RL training details.

D.1 Environments and tasks

The renders of the environments used in our experiments can be seen in Figure 8. Note, that the only difference between *FetchPickAndPlace-v4* and *FetchPush-v4* is in disabling the end-effector gripper with the latter. This corresponds to the object goal positions being directly on top of the table, whereas with *FetchPickAndPlace-v4*, the goal positions can be up to 45 centimeters high. In all *Fetch* task variants the object can be moved off the table potentially leading to its free-fall. This is the reason for elevated $Door(s)$ values in Figure 3 at larger distances.

D.2 Heuristic learning

Obtaining all heuristics in our experiments is conducted by using a random policy for a predefined number of episodes. In addition, we also allow initialization of the environment to any joint (*Pendulum* and *Fetch* environments) or Cartesian position *PointMaze* and velocity. We choose this episode initialization scheme to allow real-world *Pendulum* and *Fetch* applicability, while maintaining relatively low number of samples needed for convergence of heuristics. Once the heuristics converge, however, their values are the same regardless of the choice of the initial state distribution. We list the hyperparameter values in Table 2. Note the robustness of the hyperparameters across tasks and methods. The details specific to individual heuristics are below.

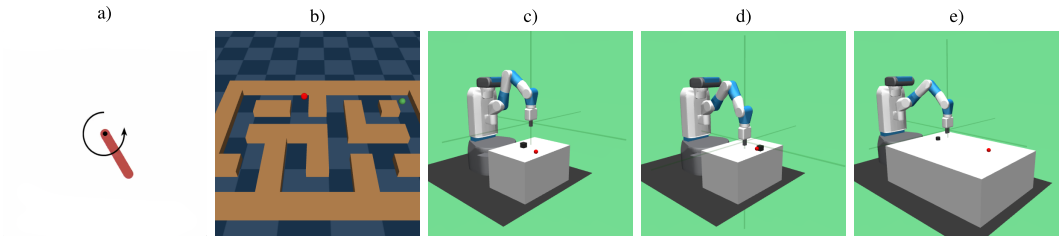

Figure 8: Renders of the environments used in our experiments. a) *Pendulum-v1*, b) *PointMaze_Large-v3*, c) *FetchPickAndPlace-v4*, d) *FetchPush-v4*, and e) *FetchSlide-v4*.

Table 2: Heuristic reward model hyperparameters grouped by method.

Hyperparameter	*Pendulum*	*PointMaze*	*Fetch*
Door(s)			
horizon H	200	500	32
batch size	1024	1024	1024
learning rate	10^{-3}	10^{-3}	10^{-3}
number of Gaussian components	3	3	3
hidden layer dim.	512	512	512
μ and Σ estimator hidden layers	3	3	3
α estimator hidden layers	3	3	3
Empowerment			
horizon H	200	500	32 / 8
learning rate	10^{-3}	10^{-3}	10^{-3}
batch size	1024	1024	1024
water filling power	1.0	1.0	1.0
water filling iterations	50	50	50
water filling tolerance	10^{-4}	10^{-4}	10^{-4}
A matrix estimator hidden dim.	512	512	512
A matrix estimator hidden layers	3	3	3
bias hidden layer dim.	256	256	256
bias estimator hidden layers	3	3	3
Count-Based			
learning rate	-	10^{-3}	10^{-3}
batch size	-	1024	1024
MDMA layer depth	-	3	3
MDMA layer width	-	3	3
MDMA model width	-	1000	1000
DIAYN			
learning rate	-	-	10^{-3}
batch size	-	-	1024
discriminator hidden dim.	-	-	512
discriminator hidden layers	-	-	3

Table 3: TQC RL algorithm hyperparameters for *Pendulum*, *PointMaze*, and *Fetch* environments.

Hyperparameter	*Pendulum*	*PointMaze*	*Fetch*
learning rate	10^{-3}	10^{-3}	10^{-3}
buffer size	50k	1M	1M
batch size	256	1024	1024
target critic soft update τ	0.005	0.005	0.05
discount factor γ (pretraining)	0.99	0.99	0.9
discount factor γ (downstream learning)	0.99	0.99	0.95
gradient steps (per env. step)	1	1	1
entropy coef.	auto	auto	auto
drop top k quantiles	$k = 2$	$k = 2$	$k = 2$

Door(s). We select the number of Gaussian components to strike a balance between the model expressivity and dimensionality of the outputs of the μ, Σ and α estimators. We find, however, that even using only a single Gaussian produces meaningful results.

Empowerment. The approach for computation of the empowerment presented by [10] obtains its values through Gaussian channel capacity. We follow their implementation for approximation of the Gaussian channel:

$$Y = AX + b \tag{9}$$

Above is the general equation, but in the empowerment case $Y = s_{t+H}$ and $X = a_{t\ldots t+H}$, i.e. Y is the observed state after H time-steps, and X is the sequence of actions, respectively. The capacity of the Gaussian channel can then be computed using singular value decomposition of the matrix A. Matrix A and vector b are approximated with fully connected neural networks receiving current observation s_t. We use an action encoder for horizons $H \in \{32, 200, 500\}$ with three 512-dimensional fully connected hidden layers. Original paper approximates empowerment online by collecting states and simultaneously unrolling H steps in a cloned separate environment with a random policy. The reliance on the random policy represents a similar constraint as the one imposed by the heuristic pretraining procedure employed in this paper.

Count-based. We implement the count-based approach following the theory from [5], but to extend it to continuous state spaces, we use a probability density estimator. We tested several methods, including normalizing flow models, and selected the Marginalizable Density Model Approximator (MDMA) [42] for its superior robustness across diverse testing environments.

DIAYN. To use the entropy-maximization objective, we pretrain the discriminator $D(z \mid s)$. The entropy of the skill likelihoods z given the state s is then used as a reward for the RL agent. In the original paper, the discriminator is trained simultaneously with the policy, but in this work we employ the same training scheme as with other heuristics. Pretraining under random behavior policy gives us similar results to [10], where this approach is used as one of the baselines.

D.3 Reinforcement learning

Reinforcement learning algorithm hyperparameters are listed in Table 3. The policy π and critics Z are approximated with fully connected neural networks. We also use Hindsight Experience Replay (HER) [43] with *Fetch* tasks to accelerate generalization to arbitrary object initial and target positions. We use a $1:4$ ratio between real and virtual samples.