Mixture of Autoencoder Experts Guidance using Unlabeled and Incomplete Data for Exploration in Reinforcement Learning

Anonymous authors
Paper under double-blind review

Keywords: Reinforcement Learning, Intrinsic Motivation, Expert Demonstrations, Incomplete Data, and Exploration.

Summary

Recent trends in Reinforcement Learning (RL) highlight the need for agents to learn from reward-free interactions and alternative supervision signals, such as unlabeled or incomplete demonstrations, rather than relying solely on explicit reward maximization. Developing generalist agents that can adapt efficiently in real-world environments often requires leveraging these reward-free signals to guide learning and behavior. While intrinsic motivation techniques provide a means for agents to seek out novel or uncertain states in the absence of explicit rewards, they are often challenged by dense reward environments or the complexity of high-dimensional state and action spaces. Furthermore, most existing approaches rely directly on the unprocessed intrinsic reward signals, which can make it difficult to shape or control the agent's exploration effectively. We propose an approach that can effectively utilize expert demonstrations, even when they are incomplete and imperfect. By applying a mapping function to transform the similarity between an agent's state and expert data into a shaped intrinsic reward, our method allows for flexible and targeted exploration of expert-like behaviors. We employ a Mixture of Autoencoder Experts to capture a diverse range of behaviors and accommodate missing information in demonstrations. Experiments show our approach enables robust exploration and strong performance in both sparse and dense reward environments, even when demonstrations are sparse or incomplete. This provides a practical framework for RL in realistic settings where optimal data is unavailable and precise reward control is needed.

Contribution(s)

1. This paper introduces MoE-GUIDE, an RL framework that learns from incomplete, unlabeled, and imperfect expert demonstrations by using a Mixture of Autoencoders as a similarity model.
   Context: Prior work on leveraging demonstrations in RL typically assumes complete and high-quality demonstrations, making them less applicable to realistic scenarios with partial or noisy data.
2. We propose a mapping function that transforms state similarity with expert data into a shaped intrinsic reward, enabling flexible and targeted exploration.
   Context: Existing intrinsic motivation approaches often rely on unprocessed intrinsic rewards, which can make exploration hard to control or insufficiently focused.
3. We demonstrate that MoE-GUIDE enables robust exploration and strong performance in both sparse and dense reward environments, even when expert data is limited, incomplete, or partially observed.
   Context: Prior methods tend to degrade in performance when provided with sparse or imperfect demonstrations, whereas our method maintains effectiveness across a range of challenging settings.

Mixture of Autoencoder Experts Guidance using Unlabeled and Incomplete Data for Exploration in Reinforcement Learning

Anonymous authors
Paper under double-blind review

Abstract

Recent trends in Reinforcement Learning (RL) highlight the need for agents to learn from reward-free interactions and alternative supervision signals, such as unlabeled or incomplete demonstrations, rather than relying solely on explicit reward maximization. Developing generalist agents that can adapt efficiently in real-world environments often requires leveraging these reward-free signals to guide learning and behavior. While intrinsic motivation techniques provide a means for agents to seek out novel or uncertain states in the absence of explicit rewards, they are often challenged by dense reward environments or the complexity of high-dimensional state and action spaces. Furthermore, most existing approaches rely directly on the unprocessed intrinsic reward signals, which can make it difficult to shape or control the agent's exploration effectively. We propose an approach that can effectively utilize expert demonstrations, even when they are incomplete and imperfect. By applying a mapping function to transform the similarity between an agent's state and expert data into a shaped intrinsic reward, our method allows for flexible and targeted exploration of expert-like behaviors. We employ a Mixture of Autoencoder Experts to capture a diverse range of behaviors and accommodate missing information in demonstrations. Experiments show our approach enables robust exploration and strong performance in both sparse and dense reward environments, even when demonstrations are sparse or incomplete. This provides a practical framework for RL in realistic settings where optimal data is unavailable and precise reward control is needed.

1 Introduction

The pursuit of intelligent, adaptive agents in reinforcement learning (RL) increasingly requires methods that go beyond traditional reward maximization. In many real-world settings, agents must learn and generalize from limited or ambiguous feedback, such as sparse environmental rewards, incomplete demonstrations, or unlabeled experience. These scenarios highlight the need for RL approaches that can leverage alternative signals, whether from the environment or from human guidance, to form robust representations and discover useful behaviors.

A key strategy for enabling learning in such settings has been the use of intrinsic motivation, which encourages agents to seek out novel, uncertain, or otherwise informative states. Techniques such as curiosity-driven exploration Pathak et al. (2017) and Random Network Distillation (RND) Burda et al. (2018) provide intrinsic rewards based on prediction errors or novelty estimates, guiding agents through unfamiliar regions of the state space. More recently, autoencoder-based methods have emerged as powerful tools for quantifying state familiarity, using reconstruction loss to identify and reward visits to underexplored or novel states Klissarov et al. (2019); Kubovčík et al. (2023); Yan et al. (2024); Liu et al. (2019). While these approaches have demonstrated effectiveness in sparse-reward environments, they depend critically on learning meaningful representations, which can be challenging in high-dimensional, continuous environments Aubret et al. (2019).

Demonstrations can hold valuable information, and therefore, approaches like Behavior Cloning Pomerleau (1989) directly imitate expert trajectories but lack the flexibility to improve beyond sub-optimal data. Inverse reinforcement learning (IRL) Ng et al. (2000) and guided exploration methods Ho & Ermon (2016) infer reward functions or policies from demonstrations, allowing for some autonomy; however, they most of the time do not utilize extrinsic rewards and struggle to outperform the expert. Demonstrations are also used in RL to improve performance by adding them to the replay buffer Paine et al. (2019); Rajeswaran et al. (2017) or by leveraging BC to jump-start the policy or as a guide during the training process Paine et al. (2019); Rajeswaran et al. (2017). Importantly, these approaches typically assume access to complete trajectory data, including actions and next states. In practice, however, obtaining such complete datasets is challenging. Technical limitations, privacy concerns, and sensor issues frequently result in missing, incomplete, or noisy demonstration data Rao et al. (2018); Zhao & Zhang (2019); Cao et al. (2023). This is common in domains like robotics and traffic modeling, where actions or states may only be partially observed or where data is sparse due to sensor failures or limited coverage Torabi et al. (2018a); Wei et al. (2020); Sun & Ma (2019); Xu et al. (2021). Collecting high-quality, dense expert data is often costly, time-consuming, or impossible, and may still yield imperfect demonstrations Camacho et al. (2021). Demonstrations may also be sparse or imperfect, such as those from low-bitrate videos. Consequently, methods that require fully observable, dense demonstrations may perform poorly in real applications. This highlights the need for research on methods that can learn effectively from incomplete, state-only, or imperfect demonstrations. This is a much more challenging but realistic scenario.

Although state-only IL and IRL methods can be used to enrich extrinsic rewards with incomplete and unlabeled data, guiding the agent to regions in the observation space, these reward models will be computationally expensive to obtain, as they require numerous interactions with the environment Zare et al. (2024). Therefore, our method utilizes expert demonstrations to train a model, such as an autoencoder, density estimator, or RND, that measures the similarity between current states and expert behavior, thereby producing a loss landscape over the observation space. We introduce a mapping function that normalizes model loss into an intrinsic reward, ranging from 0 to 1. States with losses below the minimum threshold are considered expert-like and receive the highest reward. States with losses above the maximum threshold receive zero reward. The reward is calculated using a chosen mapping function, such as linear or exponential, for losses in between. This approach enables precise control over the reward structure, allowing us to eliminate undesirable local minima and encourage exploration in regions likely to yield expert-like outcomes.

An essential aspect of this framework is the similarity model's ability to distinguish expert behavior from random states. Our research found that standard autoencoders with narrow bottlenecks can be highly selective for expert data, as they focus solely on extracting useful features to optimize reconstruction of the expert behavior. However, as in the case of autoencoders, a single similarity model may still struggle to capture the full diversity of expert demonstrations. To address this, we introduce Mixture of Experts Guidance using Unlabeled and Incomplete Data for Exploration (MoE-GUIDE), a mixture-of-experts model using several similarity models, each specializing in different features or modes of the expert data. A gating network combines its outputs, dynamically weighting each expert for a given state. This forms well-defined regions in the observation space, similar to expert-like states. By converting this landscape into an intrinsic reward, the agent is guided toward regions aligned with expert experience, thereby improving exploration efficiency.

2 Background

2.1 Reinforcement Learning Beyond Rewards

Traditional Reinforcement Learning (RL) is grounded in the Markov Decision Process (MDP) framework, where an agent interacts with an environment by observing states $s \in \mathcal{S}$, selecting actions $a \in \mathcal{A}$, and receiving rewards $r \in \mathbb{R}$ defined by a reward function $R : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ Sutton et al. (1998). The agent's objective is typically to maximize the expected cumulative discounted return:

$$\mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right], \tag{1}$$

where $\gamma \in [0, 1)$ is the discount factor.

However, real-world environments often lack well-specified, dense, or even meaningful reward signals. This has motivated a growing body of research on reward-free RL Jin et al. (2020), where agents learn from alternative forms of supervision, such as unlabeled interaction data, expert demonstrations, preferences, or implicit human feedback. Reward-free RL aims to develop agents that can acquire generalizable skills and representations from environmental structure or diverse signals, thereby facilitating rapid adaptation when task rewards become available or when rewards are difficult to specify.

2.2 Representation Learning and Intrinsic Motivation

A core challenge in reward-free RL is learning meaningful representations and skills from unlabeled data. Intrinsic motivation offers one solution, providing internal reward signals that incentivize exploration, skill development, or the acquisition of predictive representations.

Prediction- and surprise-based methods reward the agent for visiting novel or unpredictable states. The Intrinsic Curiosity Module (ICM) Pathak et al. (2017) measures the prediction error of a learned forward model as an intrinsic reward, thereby encouraging the agent to seek out transitions that are difficult to predict. Similarly, Random Network Distillation (RND) Burda et al. (2018); Yang et al. (2024) assesses state novelty by comparing the output of a fixed random network to that of a predictor network, guiding exploration toward poorly represented states.

Novelty- and count-based strategies encourage agents to visit rarely encountered states, either through explicit state visitation counts in discrete domains or via pseudo-counts and density models in high-dimensional spaces Bellemare et al. (2016); Ostrovski et al. (2017); Zhao & Tresp (2019). These approaches increase the diversity of experiences and can improve sample efficiency in sparse-reward environments.

2.3 Learning from Demonstrations and Alternative Signals

When reward functions are ill-defined or unavailable, demonstrations and other human-centric signals can serve as crucial supervisory information. Demonstration-driven techniques, such as imitation learning Ho & Ermon (2016), inverse reinforcement learning Ng et al. (2000), and learning from observation Torabi et al. (2018b); Zhu et al. (2020), leverage expert trajectories or behavioral cues to shape agent behavior. These methods can guide the learning process of an agent.

Recent advances have addressed challenges such as incomplete, suboptimal, or action-free demonstrations Wei et al. (2020); Xu et al. (2021); Camacho et al. (2021); Fu et al. (2017). However, these methods do not account for extrinsic rewards, and therefore, demonstrations have also been integrated into off-policy RL via replay buffers Paine et al. (2019); Rajeswaran et al. (2017). Other methods use hand-crafted reward terms based on demonstrations Peng et al. (2018) and leverage BC to pretrain the policy or to guide the learning process Nair et al. (2018); Rajeswaran et al. (2017). However, these methods assume complete data that exhibits near-perfect behavior. Therefore, our method proposes a framework to guide the learning process of an agent with incomplete, unlabeled, and imperfect demonstrations.

2.4 Soft Actor-Critic

Our method builds upon the Soft Actor-Critic (SAC) framework Haarnoja et al. (2018a;b), an off-policy actor-critic algorithm that augments the reward maximization objective with an entropy maximization term. This encourages diverse behavior and robust exploration:

$$J(\pi) = \mathbb{E}_{\tau \sim \rho_\pi} \left[\sum_{t=0}^{T} r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \right], \tag{2}$$

where $\mathcal{H}$ is the policy entropy and α controls the trade-off between reward and entropy. In this work, we extend SAC with intrinsic rewards derived from expert demonstrations, allowing the agent to benefit from both reward-free guidance and extrinsic rewards when available.

3 Methods

This work introduces Mixture of Experts Guidance using Unlabeled and Incomplete Data for Exploration (MoE-GUIDE), a novel method for RL that learns representations from unlabeled data while addressing the challenges posed by the limited information available in expert demonstrations, specifically the presence of gaps in the data and lack of access to actions and next states. These limitations make it infeasible to rely on the conventional techniques, such as demo replay buffers, and leveraging BC. However, expert demonstrations offer valuable insights into desirable trajectories within the environment, even if they are imperfect or limited in scope. We convert these demonstrations into an intrinsic reward, which guides the agent to regions the expert has likely visited. This intrinsic reward can be used alone or as an exploration bonus, allowing the agent to deviate from expert behavior when discovering higher extrinsic rewards. The environment's reward can prevent the agent from becoming confined to suboptimal behaviors, and a decay function can gradually reduce the influence of expert demonstrations over time. Importantly, since the intrinsic reward is a function of state only, its inclusion does not alter the set of optimal policies for the original environment reward (Ng et al., 1999). This ensures that while the agent benefits from guided exploration early in training, the optimal solution with respect to the environment's objective remains unchanged if the intrinsic reward is decayed to 0. We provide a formal argument in Appendix A.

Pretraining the agent by resetting the simulator to states from expert demonstrations exposes it to regions of the environment visited by the expert, making it easier for the agent to discover and revisit promising areas during training; however, this technique relies on the simulator supporting such resets, which may not be possible in environments with image-based observations or partial observability, and with suboptimal demonstration you may guide the agent to unwanted local minima.

In this research, we utilize autoencoders as a similarity model, as their bottleneck enables the effective detection of expert behavior, focusing solely on extracting useful features to optimize the reconstruction of expert behavior. We chose autoencoders over variational autoencoders (VAEs) because our tests showed that autoencoders were better at distinguishing expert behavior from other trajectories, while VAEs, likely due to their probabilistic nature, generalized too much and struggled to separate expert from non-expert behavior. To model expert demonstrations, we employ a mixture of autoencoder experts (MoE) as shown in Figure 1. Rather than relying on the reconstruction loss of a single autoencoder, the experts collectively reconstruct the input as accurately as possible. The MoE model includes two main components: a set of autoencoders (experts) and a gating network, which dynamically assigns a weight to each expert's output, allowing each autoencoder to specialize in distinct features or patterns of expert behavior.

The final reconstruction is computed as a weighted sum of the outputs of all active experts. For a given input x, each expert we produces a reconstruction $\text{expert}_i(x)$, and the gating network assigns a weight $\text{weight}_i(x)$ to that expert. The final reconstruction $\hat{x}$ is then given by:

$$\hat{x} = \sum_{i=1}^{N} \text{weight}_i(x) \cdot \text{expert}_i(x), \tag{3}$$

where N is the number of experts. By enabling the experts to specialize and collaborate, the MoE effectively captures the diverse characteristics of the expert behavior.

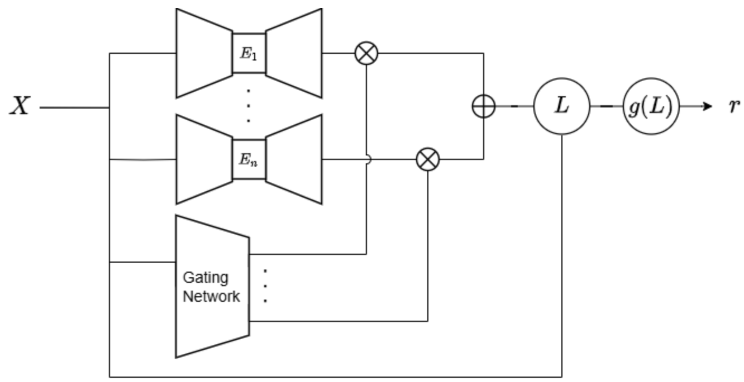


Figure 1: Diagram of the Mixture of Experts framework structure, consisting of n experts and a gating network. The gating network dynamically assigns weights to experts based on the input state X, enabling collaborative reconstruction of the input through a weighted combination of active experts' outputs. The reconstruction loss L is normalized and converted into a reward signal r by the mapping function $g(L)$, which is then used for guidance.

To guide exploration, we convert the loss induced by the similarity model into an intrinsic reward signal for the agent. Specifically, we define a mapping function g that transforms the reconstruction loss at each state, denoted by L, into a normalized reward within $[0, 1]$. This process effectively translates the structure of the loss landscape, reflecting the agent's similarity to expert-like states, into intrinsic motivations that can complement the environment's extrinsic reward.

For a given reconstruction loss L, the mapping function $g(L)$ assigns a reward of 1 when the loss is below a minimum threshold $L_{\min}$, and a reward of 0 when the loss exceeds a maximum threshold $L_{\max}$. In between the values are normalized between 0 and 1, and a monotonically increasing function $f : [0, 1] \to \mathbb{R}$ is applied, which determines how fast the rewards drop off. The mapping function is defined as

$$g(L) = \kappa \cdot \operatorname{clip}\left(f\left(\frac{L - L_{\min}}{L_{\max} - L_{\min}}\right), 0, 1\right), \tag{4}$$

where κ is a scaling factor. In this research, we use an exponential function,

$$f(x) = 1 - \exp(-sx), \tag{5}$$

where s is a steepness parameter controlling how sharply the reward increases as the loss approaches $L_{\min}$. This mapping provides high rewards for being close to $L_{\min}$ but extremely low rewards when close to $L_{\max}$.

The exploration bonus can, for example, be integrated into the Q-function update equation as follows:

$$Q'(s, a) = Q(s, a) + \alpha\left(r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right) + \beta \cdot R_{\text{sim}}(s), \tag{6}$$

where a is a given action, s is a state, α is the learning rate, γ is the discount factor, and $R_{\text{sim}}(s)$ is the intrinsic reward derived from the loss. Since $R_{\text{sim}}(s)$ stays unchanged during the training process, it can be calculated once and added to the replay buffer, limiting the computational overhead of this method. The parameter β controls the influence of the intrinsic reward and can decay over time according to a predefined schedule, such as:

$$\beta_t = \beta_0 \cdot \exp(-\lambda t), \tag{7}$$

where β_0 is the initial value of β, λ is the decay rate, and t is the training step. This decay mechanism ensures that the agent relies more heavily on the expert demonstrations during the early stages of training, gradually shifting toward autonomous learning as training progresses.

4 Experiments

We evaluate MoE-GUIDE combined with Soft Actor-Critic (SAC) on five MuJoCo continuous control benchmarks: Swimmer, Hopper, Walker2d, HalfCheetah, and Ant. Each environment is provided with a limited set of expert demonstrations: one for Swimmer, four for Hopper, and ten for Walker2d, HalfCheetah, and Ant. Demonstrations are sparsely sampled by recording states every four steps, resulting in incomplete coverage. Additional details on data collection, environment setup, and hyperparameters, as well as tables listing the final mean rewards for each figure, are provided in the Appendix.

4.1 Main Evaluation Results

We compare the following approaches using the average mean reward over 100 episodes: (1) using only extrinsic rewards (ER-only), (2) combining extrinsic rewards with pretraining on demonstration data without having guidance afterwards (ER+pretraining), (3) using the intrinsic reward from the MoE-GUIDE model with pretraining (IR+pretraining), and (4) combining extrinsic and intrinsic rewards (MoE-GUIDE), with pretraining used where applicable. For completeness, we also evaluated RND and ICM baselines; however, these methods performed poorly in dense reward environments, and their results are reported in the Appendix.

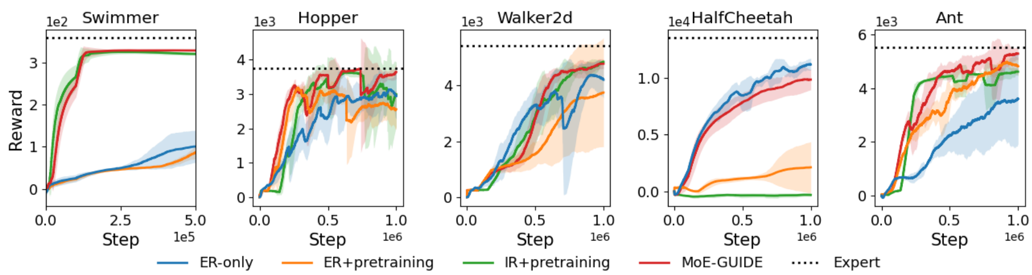


Figure 2: A comparison of (1) learning with only extrinsic rewards (ER-only), (2) combining extrinsic rewards with pretraining (ER+pretraining), (3) using only the intrinsic reward with pretraining (IR+pretraining), and (4) combining extrinsic and intrinsic rewards (MoE-GUIDE) using expert behavior that achieves high rewards.

The first experiment aimed to find demonstrations of strong experts that took longer than 1 million time steps to train, to see if guided exploration could make them perform similarly within the 1 million time steps. The results are shown in Figure 2. In Swimmer, Walker2d, and Ant, MoE-GUIDE reliably improves over using only extrinsic rewards. In Hopper, both IR-only and MoE-GUIDE reach expert-level performance during training, demonstrating that the intrinsic reward provides particularly effective guidance in this environment. By contrast, HalfCheetah has extrinsic rewards, which effectively guide the regions that yield high rewards, making it challenging to improve upon this by adding intrinsic rewards. We observe that the intrinsic reward alone, apart from HalfCheetah, is sufficient to surpass using only the extrinsic reward. However, pretraining on demonstration data does not always lead to better final performance compared to using only extrinsic rewards; once the pre-training is over, the agent may not be able to follow the expert behavior without the guidance from MoE-GUIDE.

In the HalfCheetah environment, our model recognizes expert-like behavior in the initial region and after a certain gap, but struggles in the intermediate states just beyond the start. As a result, the agent tends to remain near the initial states. Lowering the $L_{\max}$ threshold can eliminate the few well-recognized initial states, but at the cost of further widening the gap to the next expert-like region, making it more challenging to reach those regions. To address this, extra intrinsic motivation could be added to encourage the agent to explore after the initial states, such as an episodic reward to promote novel exploration, or simply an additional intrinsic reward signal.

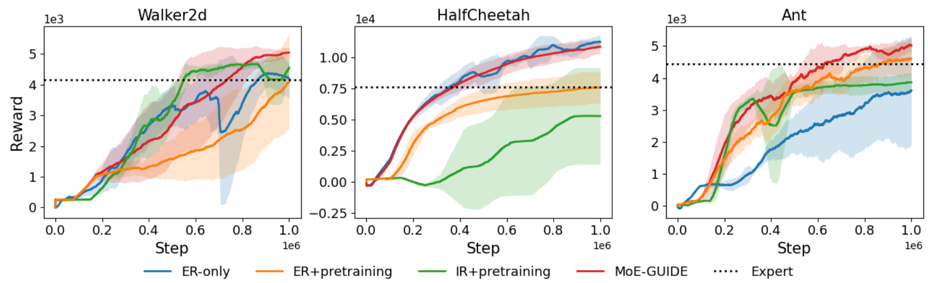


Figure 3: A comparison of (1) learning with only extrinsic rewards (ER-only), (2) combining extrinsic rewards with pretraining (ER+pretraining), (3) using only the intrinsic reward with pretraining (IR+pretraining), and (4) combining extrinsic and intrinsic rewards (MoE-GUIDE) using imperfect expert behavior.

In the second experiment, we investigated whether MoE-GUIDE can improve upon imperfect demonstrations generated by below-average SAC agents that did not get stuck in particularly bad local minima, which were trained for 1 million time steps. However, in the Halcheetah environment, a very poor performing expert was used to see if leveraging very poor demonstrations could still hold value. Here, we specifically leveraged the decaying influence of MoE-GUIDE's guidance over time. As shown in Figure 3, MoE-GUIDE improved upon the imperfect expert in all cases. However, for HalfCheetah, while MoE-GUIDE did not outperform the extrinsic reward baseline, it achieved comparable and notably more stable results.

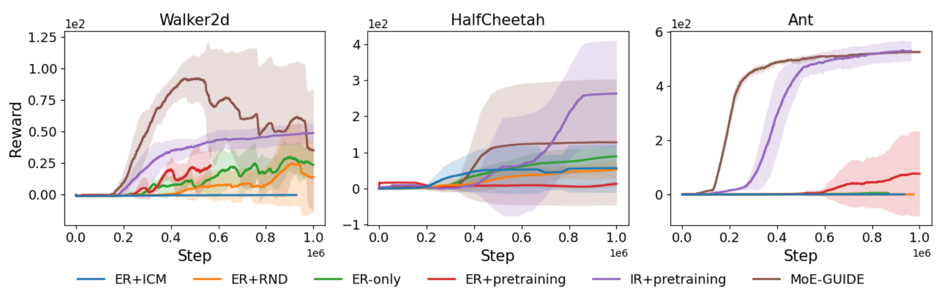


Figure 4: A comparison of (1) learning with only extrinsic rewards (ER-only), (2) learning from extrinsic rewards combined with intrinsic rewards from ICM or RND, (3) combining extrinsic rewards with pretraining (ER+pretraining), (4) using only the intrinsic reward with pretraining (IR+pretraining), and (5) combining extrinsic and intrinsic rewards (MoE-GUIDE) in a sparse partially observable environment.

Previous experiments used dense-reward environments that effectively guided agents. To create a more challenging exploration setting, MuJoCo environments were modified to provide rewards only for reaching checkpoints at fixed intervals. These intervals were chosen based on distances achievable by ICM, RND, and ER-only, since larger intervals would make it impractical to reach checkpoints within a reasonable time. Early termination results in a total reward of -1, which only applies to Walker2d and Ant, so only safe exploration is rewarded. The agent's current position is not included in the state or demonstrations, which makes the environment partially observable. As a result, agents never cross checkpoints during pretraining. Figure 4 also shows that in sparse-reward settings, MoE-GUIDE significantly outperforms baselines, even though demonstrations lack complete information about the environment. We can observe that early on, utilizing extrinsic rewards speeds up learning, but later in the training process, they can harm the learning.

4.2 Ablation Studies

This section presents an ablation study examining the impact of the gap and number of demonstrations, the $L_{\min}$ threshold, the number of experts, and different decay rates for intrinsic reward guidance on the performance of MoE-GUIDE.

4.2.1 Demonstration sparsity

We investigate the impact of demonstration sparsity by varying the number of demonstration episodes and the interval between recorded points, reducing the total number of available samples. As the demonstrations become sparser, we observe a general slight decline in agent performance; however, our method continues to provide meaningful exploration guidance even in these challenging scenarios. Notably, the agents using a single demonstration did not employ pretraining, and our approach still demonstrates robustness in this scenario, with significant gaps resulting in very few demonstration points. It can outperform baselines that rely solely on extrinsic rewards. These results highlight the effectiveness of our method in leveraging even highly limited or imperfect demonstration data to improve exploration.

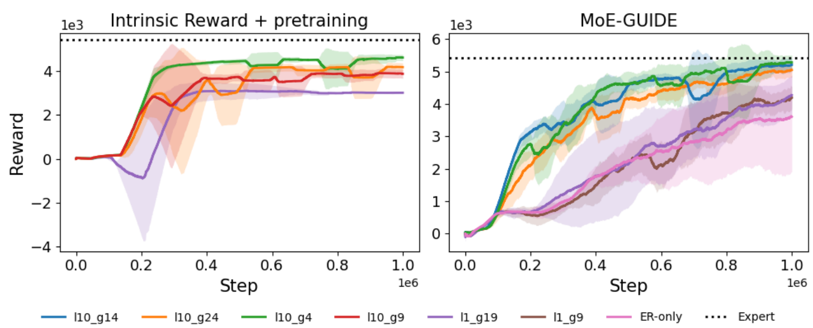


Figure 5: Demonstration size and gap robustness comparison by varying both the number of demonstration episodes provided to the model, denoted as l, and the gap parameter g, which controls how many samples are skipped between recorded demonstration points. Notably, the agents using a single demonstration did not employ pretraining.

4.2.2 Number of experts

We introduce a 3D grid world environment to visualize how our model learns from an expert path. The loss landscape is shown as in Figure 6 with heatmaps indicating the loss at fixed intervals. To highlight model behavior around the expert path, we apply a linear transformation to the loss data using a predefined $L_{\max}$. In this controlled setting, we perform ablation studies on the number of experts. The results illustrate that increasing the number of experts improves the model's ability to detect expert behavior, but also increases the risk of misclassifying other areas as expert-like. This effect is especially visible with 5 and 11 experts. Notably, the largest improvement is usually observed when increasing from 1 to 2 experts. Based on these findings, we focus on using a low number of experts in our main experiments.

4.2.3 Decay Rates

In Figure 7, we compare different decay rates for the intrinsic reward. In our implementation, the intrinsic reward is decayed **at every time step**, so the specified λ value is applied at each step, starting from 1. For Walker2d and Ant, a lower decay rate allows the expert's influence to persist longer during training, resulting in learning behavior and performance that is close to that of a standard agent. In contrast, a higher decay rate is preferred for cases such as HalfCheetah, where the expert performs significantly worse. This enables the agent to benefit from the imperfect expert primarily

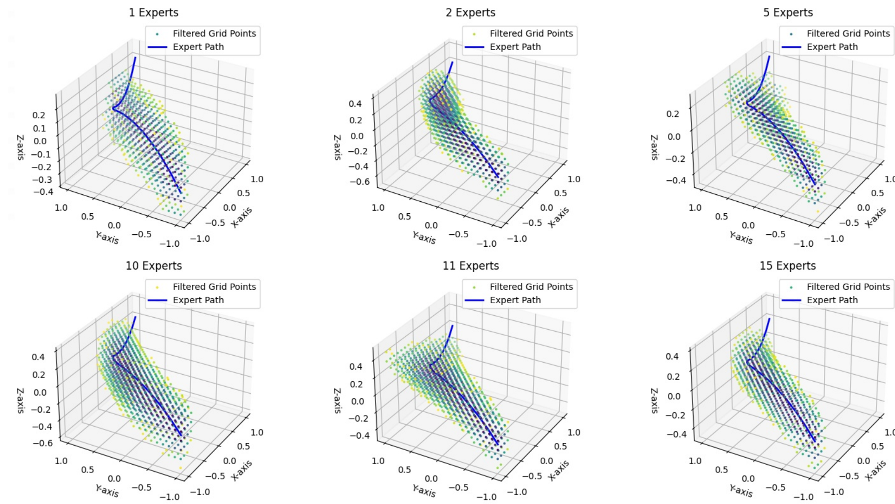


Figure 6: Visualization of the loss landscape in the 3D grid world environment. The heatmaps show the loss values at fixed intervals after applying a linear transformation with $loss_{\max}$. Results are presented for varying numbers of experts, illustrating both the enhanced capacity to identify expert behavior and the increased risk of misclassification as the number of experts increases.

in the early stages, before quickly transitioning to rely on its own learned policy. Decay rates significantly impact the agent's performance; however, by leveraging intuition about the known strength of the expert behavior, they can be estimated accurately. For instance, in the case of HalfCheetah, we knew we had a very weak expert, so high decay rates were chosen. In contrast, Walker2d performed slightly worse than an average extrinsic reward-only agent, thus requiring a low decay rate.

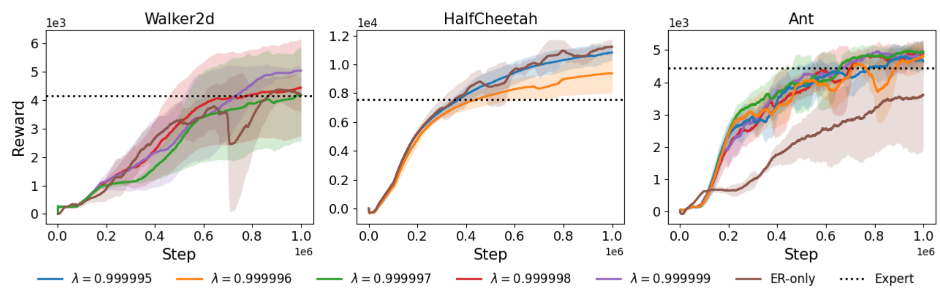


Figure 7: Learning curves for different intrinsic reward decay rates (λ). The intrinsic reward is decayed at every timestep using the specified λ value. A lower decay rate (λ closer to 1) maintains expert influence longer, while a higher decay rate quickly reduces the contribution from the expert, allowing the agent to rely more on its own learning.

4.2.4 Mapping function sensitivity

The choice of values for the mapping function plays a crucial role in the performance effectiveness and exploration of MoE-GUIDE, as demonstrated by different $L_{\min}$ values in Figure 8. Setting the threshold too high (e.g., 0.3) causes the agent to interpret too many states as expert-like, resulting in suboptimal behavior and lower extrinsic rewards while almost reaching the maximum intrinsic reward possible. In our experiments, lower values such as 0.01, 0.008, and 0.006 all result in strong extrinsic performance, with the lowest value providing the best results. However, setting $L_{\min}$ too low can also be detrimental, as the agent is unable to find the expert-like regions. For instance, during pretraining, the agent can still locate some expert-like regions, but after pretraining, it is unable to reach them on its own. The choice of values also depends on whether pretraining is applied. As with pretraining, the expert-like regions can be narrowed down since the agent visits them during the pretraining process. Therefore, without pretraining, the expert-like regions must be larger for the agent to locate them independently. Even though the results show that the $L_{\min}$ value is very sensitive, by testing different parameters and observing how well the similarity model represents the expert data, it is easily detectable when setting $L_{\min}$ or $L_{\max}$ too low when the model is not able to represent the expert behavior, but there might also be too few experts. If random trajectories can be sampled from the environment and the model represents them well, $L_{\min}$ or $L_{\max}$ is probably too large.

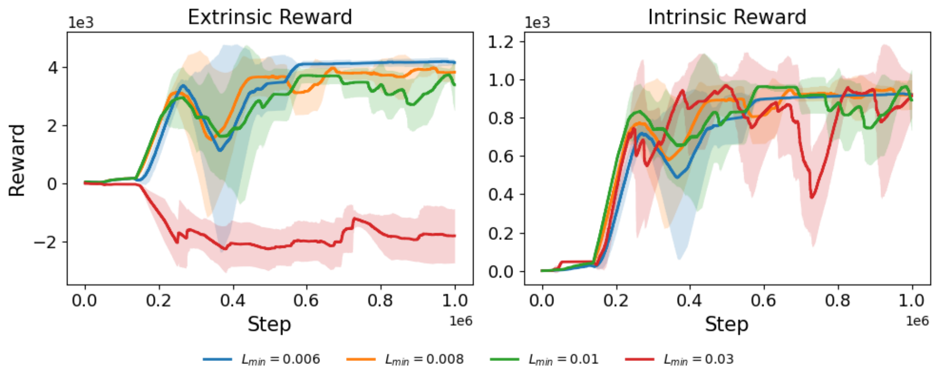


Figure 8: Impact of the mapping threshold $L_{\min}$ on MoE-GUIDE's performance. High thresholds (e.g., $L_{\min} = 0.3$) cause the agent to misidentify many states as expert-like, resulting in high intrinsic but low extrinsic rewards. Lower thresholds (0.01, 0.008, 0.006) yield better extrinsic returns, though setting $L_{\min}$ too low can make it harder for the agent to find expert-like regions after pretraining.

5 Conclusion & future works

We present MoE-GUIDE, a method for directed exploration in Reinforcement Learning that uses a mixture of similarity models trained on expert demonstrations to construct a loss landscape, which resembles the similarity of each state to expert behavior. This loss is then transformed into an intrinsic reward through a mapping function, guiding the agent towards expert-like states. MoE-GUIDE is effective in both dense and sparse reward environments, demonstrating versatility across a range of exploration challenges. The method operates successfully with demonstrations that are unlabeled, incomplete, or imperfect. Our results show that agents benefit from MoE-GUIDE, even with limited data that contains gaps. One limitation of our method is the need for manual selection of the similarity model and mapping function. In environments where extrinsic rewards already provide sufficient guidance, the additional intrinsic reward may be less beneficial and can lead to suboptimal exploration. Future work could incorporate episodic intrinsic motivation to prevent the agent from repeatedly visiting similar states and explore alternative or adaptive similarity models. In this work, we have only tested autoencoders and variational autoencoders; however, other methods, such as density estimation, ICM, and RND, could also be considered for future research. Additionally, being able to inspect the loss landscape for expert-behavior representation and misclassifications would enable more efficient and effective mapping functions.

A Properties and pitfalls of state-only intrinsic motivation

A.1 Proof that state-only intrinsic motivation does not change the optimal policy

Let $r_{\text{env}}(s,a)$ denote the environment (extrinsic) reward, and let $r_{\text{int}}(s)$ denote an intrinsic reward that depends only on the state s. The agent receives the total reward:

$$r_{\text{total}}(s,a) = r_{\text{env}}(s,a) + r_{\text{int}}(s).$$

Let $V^{\pi}_{\text{env}}(s)$ be the value function under policy π and reward r_{env}, and $V^{\pi}_{\text{total}}(s)$ under r_{total}:

$$V^{\pi}_{\text{env}}(s) = \mathbb{E}_{\left[\sum_{t=0}^{\infty} \gamma^t r_{\text{env}}(s_t,a_t) | s_0=s\right]},$$

$$V^{\pi}_{\text{total}}(s) = \mathbb{E}_{\left[\sum_{t=0}^{\infty} \gamma^t (r_{\text{env}}(s_t,a_t) + r_{\text{int}}(s_t)) | s_0=s\right]}.$$

Expanding $V^{\pi}_{\text{total}}(s)$, we get:

$$\begin{aligned} V^{\pi}_{\text{total}}(s) &= \mathbb{E}_{\left[\sum_{t=0}^{\infty} \gamma^t r_{\text{env}}(s_t,a_t) + \sum_{t=0}^{\infty} \gamma^t r_{\text{int}}(s_t) | s_0=s\right]} \\ &= V^{\pi}_{\text{env}}(s) + V^{\pi}_{\text{int}}(s), \end{aligned}$$

where

$$V^{\pi}_{\text{int}}(s) = \mathbb{E}_{\left[\sum_{t=0}^{\infty} \gamma^t r_{\text{int}}(s_t) | s_0=s\right]}.$$

The set of optimal policies under r_{env} is

$$\Pi^*_{\text{env}} = \arg \max_{\substack{\pi \\ \text{env}}(s), \quad \forall s.}$$

Under r_{total},

$$\Pi^*_{\text{total}} = \arg \max_{\substack{\pi \\ \text{total}}(s).}$$

Observation: The difference in value between any two policies π_1 and π_2 is the same under V^{π}_{env} and V^{π}_{total}:

$$V^{\pi_1}_{\text{total}}(s) - V^{\pi_2}_{\text{total}}(s) = (V^{\pi_1}_{\text{env}}(s) - V^{\pi_2}_{\text{env}}(s)) + (V^{\pi_1}_{\text{int}}(s) - V^{\pi_2}_{\text{int}}(s))$$

However, $V^{\pi}_{\text{int}}(s)$ depends only on the state visitation distribution induced by π. Since $r_{\text{int}}(s)$ does not depend on actions, optimizing $V^{\pi}_{\text{total}}(s)$ is equivalent to optimizing $V^{\pi}_{\text{env}}(s)$, as $V^{\pi}_{\text{int}}(s)$ is additive and does not affect the relative ordering of policies with respect to $V^{\pi}_{\text{env}}(s)$.

Conclusion: The set of optimal policies is unchanged. That is,

$$\Pi^*_{\text{env}} = \Pi^*_{\text{total}}.$$

Thus, adding a state-only intrinsic reward does not alter the optimal policy for the original environment reward.

This result follows the classic reward shaping theory as discussed in Ng et al. (1999). For completeness, we reproduce the argument here.

$\square$

A.2 Illustrative example of possible pitfalls

We now show an example of a pitfall of using state-only intrinsic motivation. When intrinsic rewards depend solely on visiting specific states, the agent can become overly focused on those states that provide intrinsic reward, rather than exploring the environment for potentially higher extrinsic rewards. This can lead to undesirable behavior where the agent repeatedly visits or remains within rewarding states, effectively becoming "stuck" in these regions. As a result, the agent may fail to discover more optimal strategies or reach states with significant extrinsic rewards.

If the observation includes velocity or frame-stacked states, the agent is still incentivised to move, as trying to match the velocity of the expert will encourage the agent to traverse the environment rather than remain in a single region.

Table 1 details the intrinsic and extrinsic rewards for each state in an example Markov Decision Process (MDP). The agent receives an intrinsic reward of $+1$ for visiting states S_1, S_2, and S_6. The terminal state S_6 also provides a larger extrinsic reward of $+10$. Table 2 explains the possible actions.

Figure 9 shows the transition structure of this environment. At each state, the agent can execute action a_0 (move right) or a_1 (move left). If the immediate intrinsic reward primarily drives the agent's policy, it may become trapped, oscillating between the early rewarding states (S_1, S_2, S_3), and fail to reach the high extrinsic reward at S_6.

Table 1: Intrinsic and extrinsic rewards for each state in the environment.

State	Intrinsic Reward	Extrinsic Reward
S_1	$+1$	0
S_2	$+1$	0
S_3	$+1$	0
S_4	0	0
S_5	0	0
S_6	$+1$	$+10$

Table 2: Action meanings in the MDP.

Action	Description
a_0	Move one state to the right
a_1	Move one state to the left

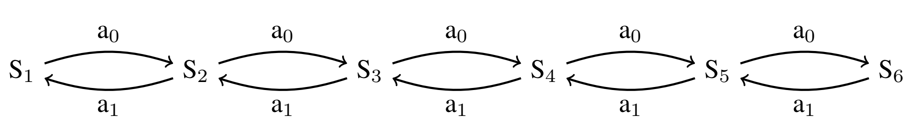


Figure 9: MDP transition graph. States S_1 to S_6 are connected by curly arrows denoting actions a_0 (right) and a_1 (left).

To mitigate this, intrinsic reward schemes can be enhanced in several ways:

- **Global intrinsic rewards decay**: After every episode or time step, β can be decayed so that the agent focuses more on maximizing the extrinsic rewards over time.
- **State/region-specific intrinsic rewards decay:** A more advanced way of decaying the intrinsic reward is to decay the intrinsic rewards for a specific state or region either within an episode

(episodic novelty) or across all episodes (lifetime novelty), reducing the incentive to revisit familiar states, while not decaying the intrinsic rewards for unvisited states.

- **Exploration Bonuses:** Methods such as Random Network Distillation (RND) or Intrinsic Curiosity Module (ICM) provide additional motivation for exploration by rewarding the agent for encountering novel or unpredictable states.

Careful design of intrinsic motivation is crucial to avoid behaviors where agents are incentivized to remain in suboptimal regions, thereby detracting from overall task performance.

B Experimental settings

B.1 Soft Actor-Critic

Table 3: Key hyperparameters for Soft Actor-Critic (SAC) in RLlib. All settings are default unless noted. Replay buffer size for Swimmer is 100,000; for all other environments, it is 1,000,000.

Parameter	**Value**
Discount factor (γ)	0.99
Actor learning rate	0.0003
Critic learning rate	0.0003
Entropy learning rate	0.0003
Optimizer	Adam
Target smoothing coefficient (τ)	0.005
Target network update frequency	0
Replay buffer size	1,000,000 (100,000 for Swimmer)
Batch size	256
Number of hidden layers	2
Hidden layer size	256
Activation function	ReLU
Target entropy	auto
N-step returns	1
Action normalization	True

B.2 Mixture of autoencoders hyperparameters and training details

Tables 4–9 present all the hyperparameters for the models used in Section 4 of this paper. These tables comprehensively document the configuration for each environment and model variant (Table 4), the parameters for the sparse environment (Table 5), the decay parameter settings (Table 6), additional configurations for iterative mask pruning (IMP) experiments (Table 7), the loss values explored (Table 8), and ablations on gap hyperparameters (Table 9).

All models are trained using the Adam optimizer with a learning rate of 0.001 for 3000 epochs. The choice of 3000 training epochs is motivated by the need to balance the models' ability to closely fit the expert demonstrations with their ability to generalize to unseen states. We observed that increasing the number of training epochs consistently decreased the reconstruction error on the training data; however, excessively long training can reduce the model's ability to generalize, as it may overfit to the expert data. Thus, 3000 epochs were selected as a compromise between accurate representation of the expert trajectories and generalization performance.

B.3 RND & ICM

For our experiments involving intrinsic motivation, we implemented Intrinsic Curiosity Module (ICM) and Random Network Distillation (RND) as auxiliary reward signals. The design and hyper-

Table 4: Hyperparameter configurations for the perfect agents experiment for each environment and model variant.

Environment	Model	Bottleneck	Num Experts	$\mathbf{L_{min}}$	$\mathbf{L_{max}}$	Mapping Function	Steepness	Scale factor
Swimmer	MoE-GUIDE	3	1	0.01	0.1	Exponential	20	1
	IR+pretraining	3	1	0.01	0.1	Exponential	20	1
Hopper	MoE-GUIDE	4	2	0.03	0.05	Exponential	100	2
	IR+pretraining	4	2	0.03	0.05	Exponential	200	2
HalfCheetah	MoE-GUIDE	7	4	0.1	0.9	Exponential	100	1
	IR+pretraining	7	4	0.1	0.9	Exponential	200	1
Walker2d	MoE-GUIDE	7	3	0.04	0.5	Exponential	100	2
	IR+pretraining	7	3	0.04	0.5	Exponential	200	2
Ant	MoE-GUIDE	10	2	4×10^{-5}	0.1	Exponential	100	2
	IR+pretraining	10	2	4×10^{-5}	0.1	Exponential	200	2

Table 5: Hyperparameter configurations for the agents trained in the sparse environment for each environment and model variant.

Environment	Model	Bottleneck	Num Experts	$\mathbf{L_{min}}$	$\mathbf{L_{max}}$	Mapping Function	Steepness	Scale factor
HalfCheetah	MoE-GUIDE	7	4	0.1	0.9	Exponential	200	0.01
	IR+pretraining	7	4	0.1	0.9	Exponential	200	1
Walker2d	MoE-GUIDE	7	3	0.04	0.5	Exponential	100	0.01
	IR+pretraining	7	3	0.04	0.5	Exponential	200	2
Ant	MoE-GUIDE	10	2	4×10^{-5}	0.1	Exponential	200	0.01
	IR+pretraining	10	2	4×10^{-5}	0.1	Exponential	200	0.01

Table 6: Decay parameter configurations for the decay ablation study for Ant, HalfCheetah, and Walker2d.

Env	Decay	Model	Bottleneck	Experts	L_{min}	L_{max}	Map Fn	Steepness	Scale
Ant	0.999995	MoE-GUIDE	10	3	6×10^{-4}	0.01	Exp	100	5
	0.999996	MoE-GUIDE	10	3	6×10^{-4}	0.01	Exp	100	5
	0.999997	MoE-GUIDE	10	3	6×10^{-4}	0.01	Exp	100	5
	0.999998	MoE-GUIDE	10	3	6×10^{-4}	0.01	Exp	100	5
	0.999999	MoE-GUIDE	10	3	6×10^{-4}	0.01	Exp	100	5
HalfCheetah	0.999995	MoE-GUIDE	7	4	0.1	0.8	Exp	100	5
	0.999996	MoE-GUIDE	7	4	0.1	0.8	Exp	100	5
Walker2d	0.999997	MoE-GUIDE	7	4	0.03	0.5	Exp	100	5
	0.999998	MoE-GUIDE	7	4	0.03	0.5	Exp	100	5
	0.999999	MoE-GUIDE	7	4	0.03	0.5	Exp	100	5

Table 7: Hyperparameter configurations for the experiment using imperfect experts for Ant, HalfCheetah, and Walker2d.

Environment	Model	Bottleneck	Num Experts	$\mathbf{L_{min}}$	$\mathbf{L_{max}}$	Mapping Function	Steepness	Scale factor
HalfCheetah	MoE-GUIDE	7	4	0.1	0.8	Exponential	100	1
	IR+pretraining	7	4	0.1	0.8	Exponential	200	1
Walker2d	MoE-GUIDE	7	4	0.03	0.5	Exponential	100	2
	IR+pretraining	7	4	0.03	0.5	Exponential	200	2
Ant	MoE-GUIDE	10	3	6×10^{-4}	0.01	Exponential	100	1
	IR+pretraining	10	3	6×10^{-4}	0.01	Exponential	200	1

Table 8: Max loss values explored in the Ant environment, with all hyperparameter columns.

Loss Value	Model	Bottleneck	Num Experts	$\mathbf{L_{max}}$	Mapping Function	Steepness	Scale factor
0.01	IR+pretraining	6	2	0.1	Exponential	100	1
0.03	IR+pretraining	6	2	0.1	Exponential	100	1
0.006	IR+pretraining	6	2	0.1	Exponential	100	1
0.008	IR+pretraining	6	2	0.1	Exponential	100	1

Table 9: Gaps for different configurations in the Ant environment, with all hyperparameter columns.

Name	Model	Bottleneck	Num Experts	L_{min}	L_{max}	Mapping Function	Steepness	Scale factor
l10_s5	MoE-GUIDE	10	2	4×10^{-5}	0.1	Exponential	100	2
l10_s5	IR+pretraining	10	2	4×10^{-5}	0.1	Exponential	200	2
l10_s10	IR+pretraining	10	3	8×10^{-4}	0.1	Exponential	200	1
l10_s15	MoE-GUIDE	10	3	3×10^{-4}	0.05	Exponential	100	2
l10_s25	MoE-GUIDE	10	3	1×10^{-4}	0.05	Exponential	100	2
l1_s10	MoE-GUIDE	10	10	0.0001	0.08	Exponential	100	1
l1_s20	IR+pretraining	10	5	0.01	0.05	Exponential	200	1

parameter selection for these methods was guided by the original works Pathak et al. (2017); Burda et al. (2018), as well as by Yuan et al. (2024), which provides extensive discussion of architectural choices, normalization strategies, and practical recommendations. In particular, Yuan et al. (2024) informed our choices of network size, orthogonal initialization, and state normalization. Additionally, following the findings of Li et al. (2019), we use only the forward model in off-policy settings, as it was shown to be sufficient for effective curiosity-driven exploration.

The following choices and hyperparameters were used consistently for both ICM and RND:

- **State Normalisation:** All states were normalised before being fed to the intrinsic modules.
- **Reward Normalisation:** Intrinsic rewards were normalised online using a running mean and standard deviation (RMS).
- **Optimizer:** Adam optimizer with a learning rate of 0.0003.
- **Feature Dimension:** 128-dimensional feature space for the learned or random embeddings.
- **Hidden Dimension:** All multi-layer perceptrons (MLPs) used in the intrinsic modules had hidden layers of size 256.
- **Weight Initialization:** All networks were initialized using orthogonal initialization.
- β**:** was chosen to provide a small exploration bonus of around 1% of the extrinsic reward.

C Data Source Selection Rationale

The selection of data sources for both the perfect and imperfect experts was guided by the need to provide a comprehensive evaluation of the agent's learning capabilities under varying supervision qualities. For the **perfect experts**, we prioritized the best available baselines, ensuring that the guidance provided to the agents represented near-optimal behavior. This allowed for a stringent assessment of whether agents could, with the aid of such expert data, achieve or approximate top-tier performance within a limited training budget of 1 million timesteps.

For the **imperfect experts**, we aimed to supply the agents with guidance that, while informative, did not represent optimal behavior. Specifically, we selected agents that performed slightly below the average Soft Actor-Critic (SAC) agent but were not trapped in poor local optima, thereby providing learning signals that were sub-optimal yet still constructive. An exception was made for the HalfCheetah environment, where we intentionally included a particularly poorly performing agent. This choice was made to test whether even data from significantly subpar agents can be rigorously could contribute positively to the learning process when combined with the proposed guidance.

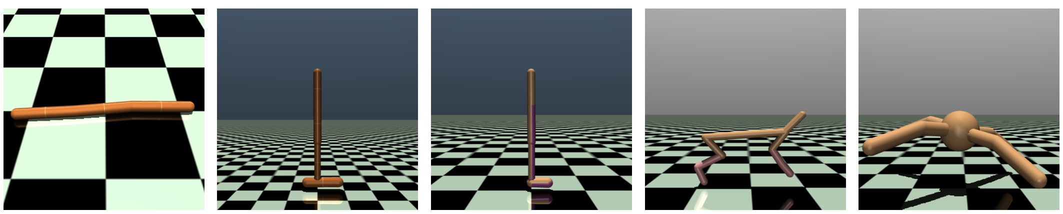
Together, these selections facilitate a thorough investigation into the robustness and effectiveness of the learning algorithms when exposed to both high-quality and imperfect supervision.

Table 10: Data sources used for training models, categorized by expert quality.

Category	Environment	Source/Description
Perfect Expert	Swimmer	Standard settings Open-Loop Baseline Raffin et al. (2023)
	Hopper	CILO paper dataset Gavenski et al. (2024)
	Walker2d	Good performing SAC agent after 2M timesteps
	HalfCheetah	Good performing SAC agent after 2M timesteps
	Ant	CILO paper dataset Gavenski et al. (2024)
Imperfect Expert	Walker2d	Below average SAC agent after 1M timesteps
	HalfCheetah	CILO paper dataset Gavenski et al. (2024)
	Ant	Slightly below average performing SAC agent after 1M timesteps that is not stuck in a bad local minima

D Environment details

We employ five standard continuous control environments from the MuJoCo suite: Swimmer, Hopper, Walker2d, HalfCheetah, and Ant. These environments are widely used to benchmark reinforcement learning algorithms in simulated robotic locomotion. In Table 11, details about the action and state spaces can be found, and Figure 10 visualizes the environments.



(a) Swimmer (b) Hopper (c) Walker2d (d) HalfCheetah (e) Ant

Figure 10: Screenshots of the MuJoCo environments used as baselines for locomotion experiments.

Table 11: Observation and action space dimensions for MuJoCo environments. Here, $\mathcal{S}$ denotes the state (observation) space and $\mathcal{A}$ denotes the action space.

Environment	$\dim(\mathcal{S})$	$\dim(\mathcal{A})$
Swimmer-v2	8	2
Hopper-v2	11	3
Walker2d-v2	17	6
HalfCheetah-v2	17	6
Ant-v2	111	8

For our experiments requiring sparse rewards, we modified the MuJoCo environments so that agents receive rewards only upon reaching checkpoints at fixed intervals, 2.5 for Walker2d, 5 for Ant, and 11 for HalfCheetah. These intervals were chosen to ensure that checkpoints are reachable, while still presenting a challenging exploration problem for the agent. Early episode termination results in a total reward of -1, incentivizing safe and deliberate exploration. HalfCheetah does not terminate early so this makes the exploration easier.

In the sparse-reward setup, the agent's position is not included in the state or demonstrations, making the environment partially observable. Additionally, the environment cannot be reset to the agent's exact demonstration location; instead, the agent is respawned at the initial starting point but initialized with the expert's joint angles and velocities. This setup exposes the agent to expert behavior without providing immediate extrinsic rewards.

E Traditional intrinsic reward methods baseline

In this section, we investigate the impact of various intrinsic motivation methods, using both standard and normalized rewards. Specifically, Intrinsic Curiosity Module (ICM), Random Network Distillation (RND), and an autoencoder-based intrinsic reward, on agent performance in a dense reward environment. While these intrinsic rewards are designed to encourage exploration in sparse settings, we observe in Figure 11 that in environments with dense rewards, they can hinder the learning process. In Halcheetah, ER+ICM has a strange shape, but this is due to the highly unstable learning curves of all the runs.

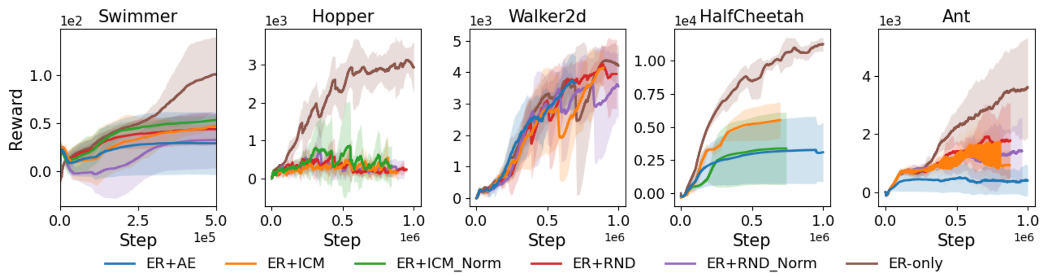


Figure 11: Performance comparison of agents trained with standard intrinsic rewards (ICM, RND, and autoencoder-based) and normalized rewards versus extrinsic rewards in a dense environment.

References

Arthur Aubret, Laetitia Matignon, and Salima Hassas. A survey on intrinsic motivation in reinforcement learning. *arXiv preprint arXiv:1908.06976*, 2019.

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.

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

Alberto Camacho, Izzeddin Gur, Marcin Lukasz Moczulski, Ofir Nachum, and Aleksandra Faust. Sparsedice: Imitation learning for temporally sparse data via regularization. In *ICML 2021 Workshop on Unsupervised Reinforcement Learning*, 2021.

Qi Cao, Yue Deng, Gang Ren, Yang Liu, Dawei Li, Yuchen Song, and Xiaobo Qu. Jointly estimating the most likely driving paths and destination locations with incomplete vehicular trajectory data. *Transportation Research Part C: Emerging Technologies*, 155:104283, 2023.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Nathan Gavenski, Juarez Monteiro, Felipe Meneguzzi, Michael Luck, and Odinaldo Rodrigues. Explorative imitation learning: A path signature approach for continuous environments. In *ECAI 2024*, pp. 1551–1558. IOS Press, 2024.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018a.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018b.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

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

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, pp. 16–32, 2019.

Martin Kubovčík, Iveta Dirgová Luptáková, and Jiří Pospíchal. Signal novelty detection as an intrinsic reward for robotics. *Sensors*, 23(8):3985, 2023.

Boyao Li, Tao Lu, Jiayi Li, Ning Lu, Yinghao Cai, and Shuo Wang. Curiosity-driven exploration for off-policy reinforcement learning methods. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1109–1114. IEEE, 2019.

Fangchen Liu, Zhan Ling, Tongzhou Mu, and Hao Su. State alignment-based imitation learning. *arXiv preprint arXiv:1911.10947*, 2019.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 6292–6299. IEEE, 2018.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Georg Ostrovski, Marc G Bellemare, Aäron Oord, and Rémi Munos. Count-based exploration with neural density models. In *International conference on machine learning*, pp. 2721–2730. PMLR, 2017.

Tom Le Paine, Caglar Gulcehre, Bobak Shahriari, Misha Denil, Matt Hoffman, Hubert Soyer, Richard Tanburn, Steven Kapturowski, Neil Rabinowitz, Duncan Williams, et al. Making efficient use of demonstrations to solve hard exploration problems. *arXiv preprint arXiv:1909.01387*, 2019.

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

Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)*, 37(4):1–14, 2018.

Dean A. Pomerleau. *ALVINN: an autonomous land vehicle in a neural network*, pp. 305–313. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989. ISBN 1558600159.

Antonin Raffin, Olivier Sigaud, Jens Kober, Alin Albu-Schäffer, João Silvério, and Freek Stulp. An open-loop baseline for reinforcement learning locomotion tasks. *arXiv preprint arXiv:2310.05808*, 2023.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017.

Wenming Rao, Yao-Jan Wu, Jingxin Xia, Jishun Ou, and Robert Kluger. Origin-destination pattern estimation based on trajectory reconstruction using automatic license plate recognition data. *Transportation Research Part C: Emerging Technologies*, 95:29–46, 2018.

Mingfei Sun and Xiaojuan Ma. Adversarial imitation learning from incomplete demonstrations. *arXiv preprint arXiv:1905.12310*, 2019.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018a.

Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018b.

Hua Wei, Chacha Chen, Chang Liu, Guanjie Zheng, and Zhenhui Li. Learning to simulate on sparse trajectory data. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 530–545. Springer, 2020.

Dayong Xu, Fei Zhu, Quan Liu, and Peiyao Zhao. Arail: Learning to rank from incomplete demonstrations. *Information Sciences*, 565:422–437, 2021.

Renye Yan, You Wu, Yaozhong Gan, Yunfan Yang, Zhaoke Yu, Zongxi Liu, Xin Zhang, Ling Liang, and Yimao Cai. Autoencoder reconstruction model for long-horizon exploration. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2024.

Kai Yang, Jian Tao, Jiafei Lyu, and Xiu Li. Exploration and anti-exploration with distributional random network distillation. *arXiv preprint arXiv:2401.09750*, 2024.

Mingqi Yuan, Roger Creus Castanyer, Bo Li, Xin Jin, Wenjun Zeng, and Glen Berseth. Rlexplore: Accelerating research in intrinsically-motivated reinforcement learning. *arXiv preprint arXiv:2405.19548*, 2024.

Maryam Zare, Parham M. Kebria, Abbas Khosravi, and Saeid Nahavandi. A survey of imitation learning: Algorithms, recent developments, and challenges. *IEEE Transactions on Cybernetics*, 54(12):7173–7186, 2024. DOI: 10.1109/TCYB.2024.3395626.

Rui Zhao and Volker Tresp. Curiosity-driven experience prioritization via density estimation. *arXiv preprint arXiv:1902.08039*, 2019.

Shuaidong Zhao and Kuilin Zhang. A distributionally robust optimization approach to reconstructing missing locations and paths using high-frequency trajectory data. *Transportation Research Part C: Emerging Technologies*, 102:316–335, 2019.

Zhuangdi Zhu, Kaixiang Lin, Bo Dai, and Jiayu Zhou. Off-policy imitation learning from observations. *Advances in neural information processing systems*, 33:12402–12413, 2020.

Supplementary Materials

The following content was not necessarily subject to peer review.

F Grid world

We present qualitative results in a gridworld with random walls, where the agent can move in any direction. The agent always selects randomly among actions that yield the highest intrinsic reward. For our method (MoE-GUIDE), intrinsic rewards are only given once per state to prevent the agent from getting stuck revisiting the same locations. In all visualizations, the green star indicates the start state, the red star marks the goal, and blue dots represent demonstration states.

Figure 12 visualizes the exploration patterns produced by different intrinsic motivation methods: random, count-based, ICM, RND, and MoE-GUIDE. This comparison highlights the distinct behaviors and exploration strategies induced by each method.

Figure 13 compares MoE-GUIDE's behavior under different amounts of missing data in the demonstrations. Each subplot corresponds to a different gap size G, where $G =$ NUMBER indicates the number of samples skipped between demonstration points. It is clearly visible that the agent attempts to explore regions where demonstration states are present, even when the demonstration data is sparse due to gaps.

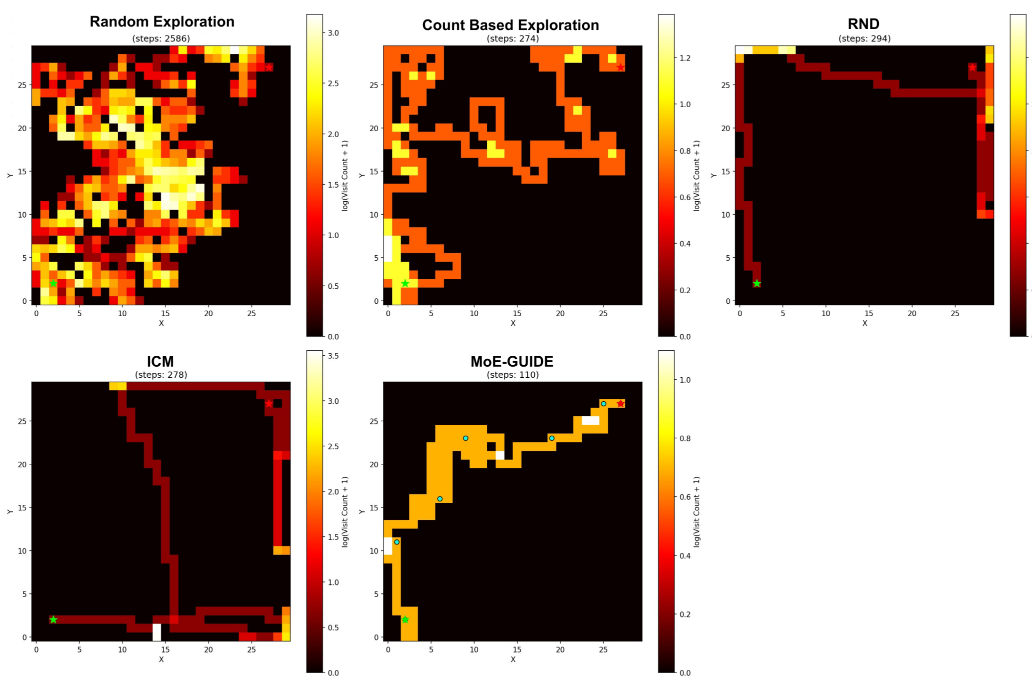


Figure 12: Exploration patterns in a gridworld with random walls for different intrinsic motivation methods: random, count-based, ICM, RND, and MoE-GUIDE. The agent always chooses among actions with the highest intrinsic reward, illustrating the characteristic exploration behavior of each method. For MoE-GUIDE, intrinsic rewards are only provided once per state to prevent the agent from getting stuck. The green star is the start state, the red star is the goal, and blue dots indicate demonstration states.

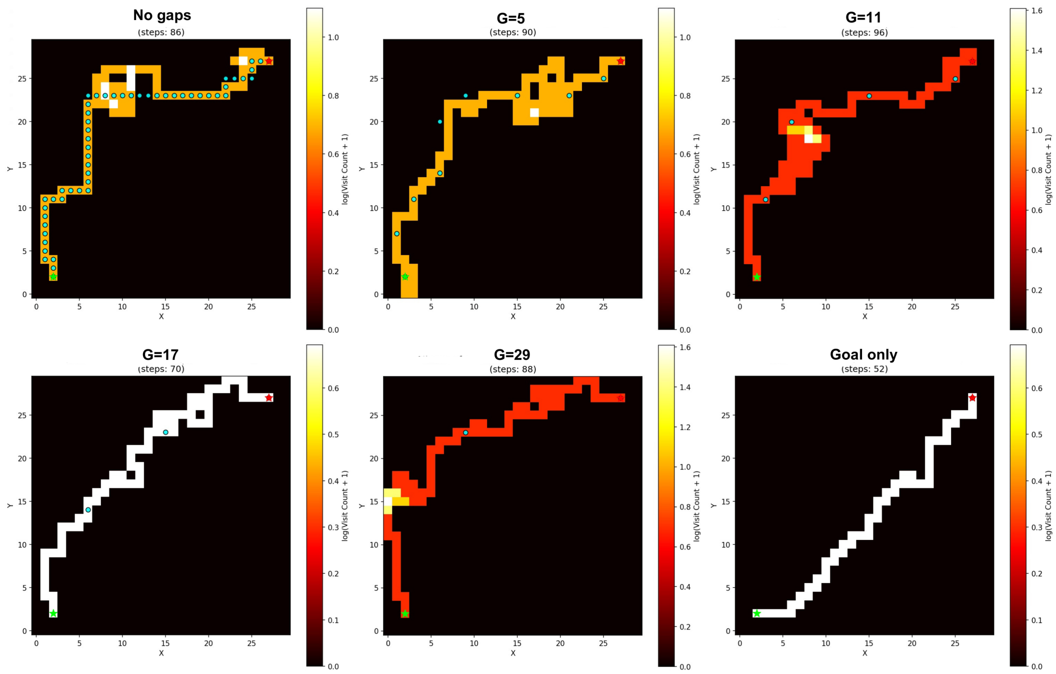


Figure 13: Effect of gaps in demonstration data on MoE-GUIDE's exploration in gridworld. Each subplot corresponds to a different gap size G, where G = NUMBER indicates the number of samples skipped between demonstration points. The agent clearly attempts to explore regions where demonstration states are present, even as the demonstration data becomes increasingly sparse. The green star is the start state, the red star is the goal, and the blue dots indicate demonstration states.

G Tables of final mean results

This section provides tables summarizing the final mean rewards and standard deviations for different experimental settings and hyperparameters. The results are presented to facilitate comparison between methods and configurations.

Table 12: Final mean rewards ± standard deviation for each method in the perfect expert experiment.

Method	**Swimmer**	**Hopper**	**Walker2d**	**HalfCheetah**	**Ant**
ER+pretraining	86.59 ± 15.25	2556.55 ± 551.80	3746.44 ± 1953.40	2112.67 ± 2250.98	4815.93 ± 324.09
IR+pretraining	321.32 ± 3.26	2994.18 ± 947.20	4841.49 ± 55.06	-325.95 ± 147.75	4611.79 ± 140.51
ER-only	100.66 ± 38.36	2946.26 ± 672.87	4201.02 ± 646.91	11216.96 ± 508.09	3603.56 ± 1704.07
MoE-GUIDE	329.50 ± 1.70	3642.78 ± 196.37	4776.34 ± 168.72	9867.41 ± 907.54	5282.29 ± 222.13

Table 13: Final mean rewards ± standard deviation for each method in the imperfect expert experiment.

Method	**Walker2d**	**HalfCheetah**	**Ant**
ER-only	4200.66 ± 646.93	11225.46 ± 505.78	3603.56 ± 1704.66
MoE-GUIDE	5046.87 ± 162.36	10829.20 ± 565.04	5020.49 ± 240.66
IR+pretraining	4557.89 ± 120.08	5274.66 ± 3884.78	3865.11 ± 279.42
ER+pretraining	4103.49 ± 1509.36	7564.65 ± 1229.63	4595.19 ± 513.56

Table 14: Final mean rewards ± standard deviation for different decay rates.

Method / Decay	Walker2d	HalfCheetah	Ant
0.999995	—	10829.20 ± 565.04	4651.73 ± 424.67
0.999996	—	9377.68 ± 1383.32	4803.90 ± 375.54
0.999997	4188.21 ± 1669.04	—	4935.50 ± 349.51
0.999998	4439.23 ± 1704.39	—	4877.52 ± 383.50
0.999999	5047.04 ± 162.74	—	4951.07 ± 121.43
ER-only	4200.63 ± 646.91	11223.27 ± 508.42	3603.59 ± 1704.69

Table 15: Final mean rewards ± standard deviation for different $L_{\min}$ values.

$L_{\min}$	Extrinsic Reward	Intrinsic Reward
0.006	4139.03 ± 93.31	919.57 ± 13.74
0.008	3812.84 ± 294.02	923.57 ± 73.03
0.01	3372.02 ± 869.04	892.51 ± 158.13
0.03	-1812.56 ± 933.01	916.76 ± 114.37

Table 16: Final mean rewards ± standard deviation for different gap sizes and number of demonstrations.

Setting	Intrinsic Reward + pretraining	MoE-GUIDE
l1_g9	—	4199.43 ± 338.67
l1_g19	3010.78 ± 34.19	4271.73 ± 615.61
l10_g4	4611.81 ± 140.51	5282.29 ± 222.68
l10_g9	3871.64 ± 203.34	—
l10_g14	—	5208.85 ± 212.66
l10_g24	4177.66 ± 191.29	5050.68 ± 225.26
ER-only		3604.23 ± 1704.66

Table 17: Final mean rewards ± standard deviations for each method in the sparse environment.

Method	Walker2d	HalfCheetah	Ant
ER-only	23.50 ± 11.85	89.26 ± 29.96	4.33 ± 8.47
ER+pretraining	23.04 ± 16.13	13.04 ± 8.31	76.39 ± 157.01
IR+pretraining	48.83 ± 7.77	263.42 ± 147.87	530.58 ± 35.52
MoE-GUIDE	34.95 ± 48.01	127.90 ± 174.62	526.62 ± 5.14
ER+RND	14.16 ± 28.72	52.79 ± 24.78	-0.23 ± 0.43
ER+ICM	-0.22 ± 0.10	56.64 ± 67.66	-0.12 ± 0.09